12

ESD-TR-83-020

Semiannual Technical Summary

Distributed Sensor Network

30 September 1982

Prepared for the Defense Advanced Research Projects Agency
under Electronic Systems Division Contract F19628-80-C-0002 by

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lexington, Massachusetts

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-80-C-0002 (ARPA Order 3345).

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Thomas J. Alpert

Thomas J. Alpert, Major, USAF
Chief, ESD Lincoln Laboratory Project Office

ADA-[redacted] 131245

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

DISTRIBUTED SENSOR NETWORKS

SEMIANNUAL TECHNICAL SUMMARY REPORT
TO THE
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

1 APRIL 1982 — 30 SEPTEMBER 1982

ISSUED 24 June 1983

Approved for public release; distribution unlimited.

LEXINGTON MASSACHUSETTS

ABSTRACT

This report describes the work performed on the DARPA Distributed Sensor Network Program at Lincoln Laboratory during the period 1 April 1982 through 30 September 1982.

CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION AND SUMMARY

The Distributed Sensor Networks (DSN) program is aimed at developing and extending target surveillance and tracking technology in systems that employ multiple spatially distributed sensors and processing resources. Such a system would be comprised of sensors, data bases, and processors distributed throughout an area and interconnected by an appropriate digital-data communication system. Surveillance and tracking of low-flying aircraft have been selected to develop and evaluate DSN concepts in the light of a specific system problem. A DSN test bed that will make use of multiple small acoustic arrays as sensors for low-flying aircraft surveillance is being developed and will be used to test and demonstrate DSN techniques and technology. This Semiannual Technical Summary (SATS) reports results for the period 1 April 1982 through 30 September 1982.

Section 2 reports on an analysis of the system characteristics of the two different acoustic target location algorithms previously developed in the DSN program. One, the reflection curve method, provides the most up to date location estimates, subject to the inherent delays of acoustic signals. It is the method now in use in the test bed. The other method, the possible position method, produces location estimates after a fixed delay from real time. A comparative analysis of the location performance of the two methods has been completed and the results are presented in Section 2. Neither method is uniformly superior but it appears that the possible position method is better than the reflection method under most circumstances. In addition, there are algorithmic complications in using the output of the reflection method resulting from sound emission time as well as target location being estimated. Subject to more analysis and experience with the present system, we are considering replacing the reflection method with the possible position method in the test bed.

Section 3 reports on acoustic data collection and signal processing activities during this reporting period. Tests were conducted to evaluate the feasibility of using remotely piloted model aircraft to provide an easier way to experiment with different scenarios and numbers of targets. The conclusion was that standard aircraft should continue to be employed as the targets for DSN experiments. Second, several approaches to characterizing the frequency content of acoustic signals have been identified and are described along with the results of some preliminary investigations. Third, experiments were conducted at Salt Lake City, Utah that validated the operational status of a mobile DSN node with its self-contained acoustically quieted generator. The test results provided verification that the generator noise was below ambient at the test site, additional knowledge concerning the orientation dependence of acoustic signatures for helicopters, and information concerning the relative performance of different microphone array configurations.

One approach in developing DSN systems is to implement them by means of autonomous cooperating processes interacting through message exchange. A major goal of the DSN project is to first validate this general approach by implementing a surveillance capability based upon it. Section 4 describes the progress made towards this goal during this reporting period. Multi-tasking facilities have been developed for nodal computers. A message based interprocess communication system has been implemented. Debugging facilities for processes running in remote nodes have been developed. These and the user interface to the system involve functions in both the remote node and in the user computer. A broadcast communication emulation has been written and is being debugged. Tracking software has been converted to operate in the nodes and to interface to the message system.

Hardware improvements and the status of the test bed nodes are summarized in Section 5. Included is a summary of progress made toward the development of an improved nodal computer configuration involving multiple processors on a single bus and containing the interface to the radio unit that will be used to provide communication and self-location functions in the test bed. Major portions of the hardware for the new nodes have been procured and a prototype is being built and tested. A prototype of the interface to the radio unit has been built and is being debugged.

Section 6 summarizes the progress made in the design and development of radio communication and internodal ranging protocols and software. It also reports on some preliminary work with Columbia University to confirm that their self-location software will be compatible with the test bed nodes.

2. ACOUSTIC LOCATION SYSTEM ANALYSIS

Two acoustic target location algorithms have been developed in the Distributed Sensor Networks program, the possible position method and the reflection method. The possible position method is described in Reference [1]. The reflection method is described in Reference [2]. Both algorithms use measured bearing data from two acoustic arrays to estimate target locations. The reflection method is the algorithm now in use in the DSN test bed.

Apart from possible differences in location performance there are two basic differences between the algorithms:

1. The possible position method requires measured bearing histories from two sensors. The reflection method requires measured bearing history from one sensor while the other sensor need only provide one measured bearing.

2. Time is an independent variable for the possible position method so that location estimates are provided for specified times. The reflection method can provide location estimates corresponding to more recent times but the time, as well as the location, must be estimated.

The reflection method seems advantageous on the surface because it requires less measured bearing data and it can provide location estimates with no more time delay than the minimum imposed by acoustic propagation times. These advantages led to its selection and use for initial experiments in the DSN test bed. But considerable tracking algorithm complexity results when the times associated with location estimates include time errors that are correlated with position errors and when the times do not constitute an ordered sequence with uniform intervals, as is the case for the reflection method. Experience has indicated that this complexity issue mitigates very much towards using the possible position method.

A comparative analysis of the target location performance of the two methods has been done as a part of an investigation to determine if the reflection method should be replaced in the test bed by the possible position method. The conclusion from this analysis is that the two methods are comparable in overall location performance and sensitivity to errors. Neither method is superior under all conditions but it appears that the possible position method actually is slightly better under a broad range of conditions and on that basis is actually preferred. The following presents more details of the results of the comparative analysis.

As a first step in comparing the target location performance of the two methods, analytical expressions were derived for the first- and second-order statistics of the errors in the location estimates produced by both methods. These statistics are functions of the sensor spacing, the target position, target bearing and speed, the statistics of bearing measurement errors, errors in assumed sensor locations and errors in the assumed velocity of sound.

The expressions for the location error statistics revealed that both methods produce unbiased location estimates if all other errors are unbiased. They also confirmed that there are certain combinations of sensor and target geometry where location inaccuracies become extremely large and revealed that both methods breakdown in identical circumstances.

Mathematically, the common condition for the failures of both methods can be stated as:

$$\sin(\Phi_a + \Phi_b) = m(\sin(\Psi_a) + \sin(\Psi_b))$$

where m is the target velocity expressed as a Mach number and where the angles are defined as per Figure 2-1. The angles Ψ_A are defined to be positive when the velocity vector is outside of the angle subtended by the line joining sensors A and B, as is the case in Figure 2-1. If m is 0, then Ψ_A and Ψ_A are defined but irrelevant. In this case, the condition reduces to the usual condition under which triangulation breaks down (i.e., when the target and sensors are collinear).

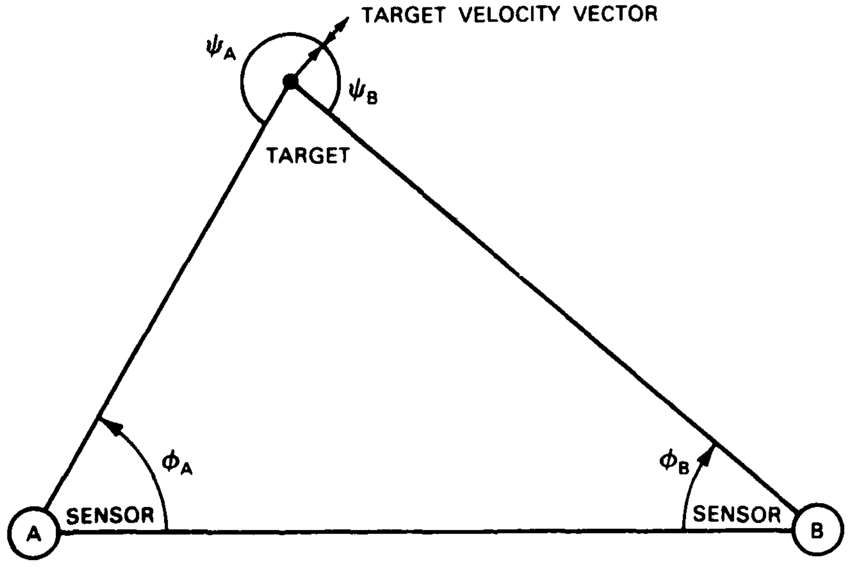


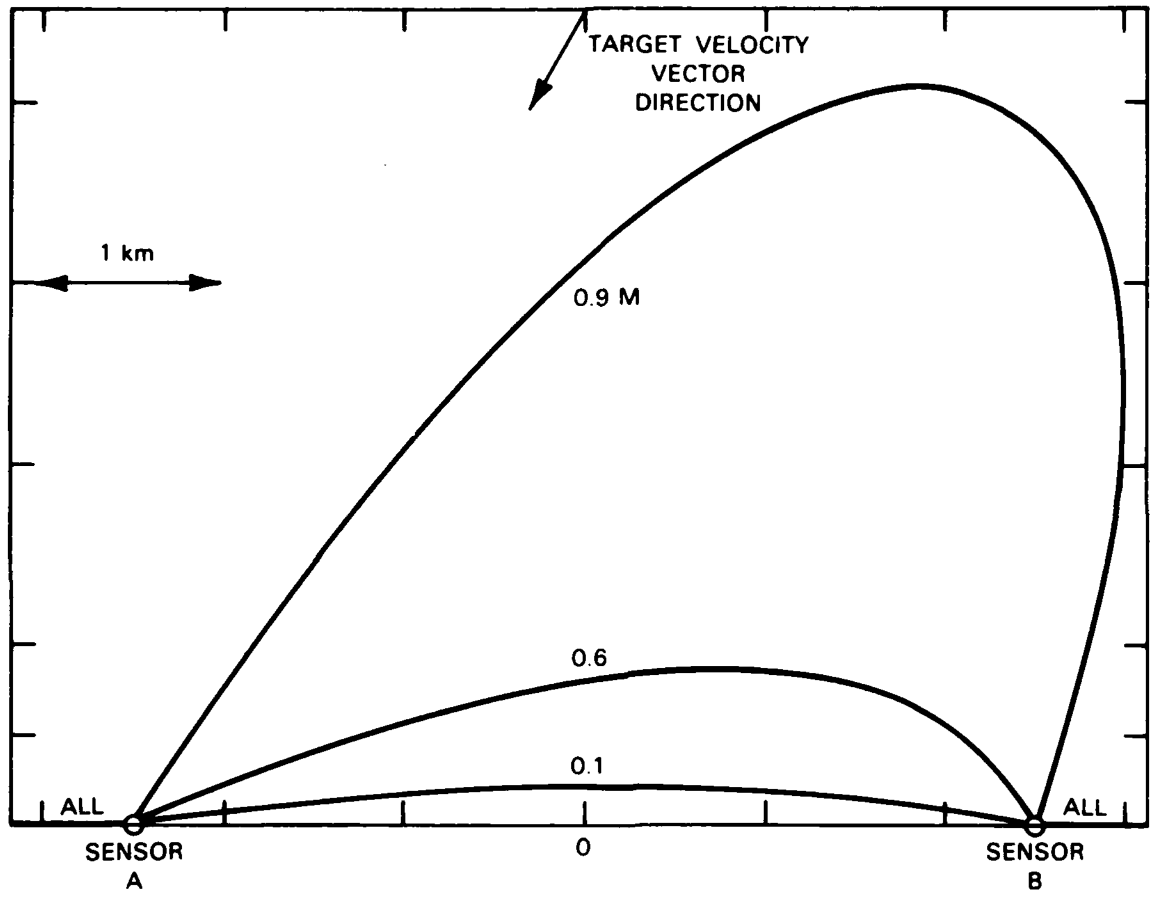
Fig. 2-1. Geometry of Target and Sensors.

Figure 2-2 plots the loci of locations at which the two methods both breakdown for one heading and three different Mach numbers. Figure 2-3 is similar but for a different heading. Note that for all targets, the methods both fail in the "end fire" geometries, (i.e., when the target and sensors are collinear but the target is not between the sensors). Only for stationary targets do the methods fail when the target is exactly between the sensors. For low target velocities, the methods breakdown in that neighborhood, but for high velocity targets the breakdown locations shift away from the region between the sensors.

The accuracy of location estimates is very sensitive to measured bearing inaccuracies, et cetera, not just on the lines plotted in the figures, but near them as well. A target with a constant heading and velocity flying between the two sensors will, as it approaches the line between the sensors, pass through a region in which it cannot be located with reasonable accuracy. A maneuvering target could pass through several such regions. This observation suggests that locations should not be calculated by the possible position method if the possible position curves are nearly tangent at their intersection or by the reflection method if the measured bearing history and the reflection curve are nearly tangent at their intersection. In effect there are velocity and heading dependent "blind spots" for location by any sensor pair. Redundant coverage is clearly necessary to compensate for this problem.

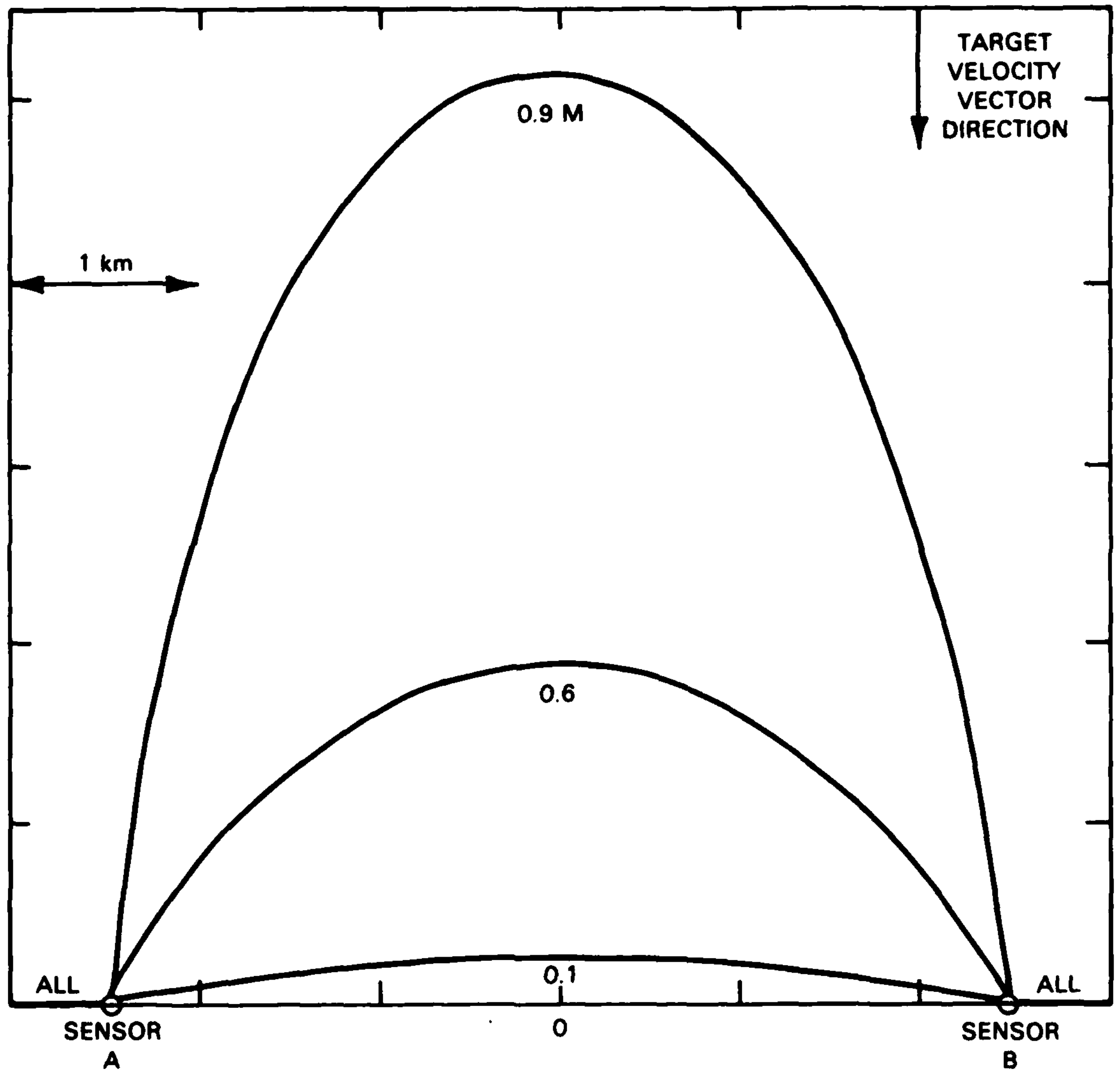
Since both location methods breakdown under the same circumstances, one might expect the accuracies of the two methods to depend identically on bearing measurement inaccuracies, et cetera. This is true only when the target is stationary or is moving along the line through the two sensors. However, the accuracy differences between the methods tend to be small if the target velocity is significantly less than the speed of sound. Figures 2-4 and 2-5 illustrate this point by showing the location accuracies of the posible position method and the reflection method for a slow moving target. Each figure shows three sensors at the vertices of an equilateral triangle 5 km on a side. The location accuracy is shown for each pair of sensors and for six target locations by plotting location accuracy circles of error probable (CEPs) about each target location for each sensor pair. Also plotted for each of the six target locations are the major and minor axes of the location accuracy ellipses for the sensor pair 1 and 2. The probability that a calculated location lies outside the appropriate ellipse is 1/e. The target is assumed to have a velocity of 0.1 Mach and a heading as indicated in the figure. The bearing measurements are assumed to have a standard deviation of 2° and are the only inaccuracies assumed.

Figures 2-4 and 2-5 are almost identical; the most noticeable difference is in the two large circles on the left side of each figure. The upper circle has a few hundred meters smaller diameter in Figure 2-4 than in Figure 2-5 while the lower circle is smaller in Figure 2-5 than



128164-N

Fig. 2-2. Target Locations Where Possible Position Method Fails (Speed of 0, 0.1, 0.6 and 0.9 Mach).



128165-N

Fig. 2-3. Target Locations Where Reflection Method Fails (Speeds of 0, 0.1, 0.6 and 0.9 Mach).

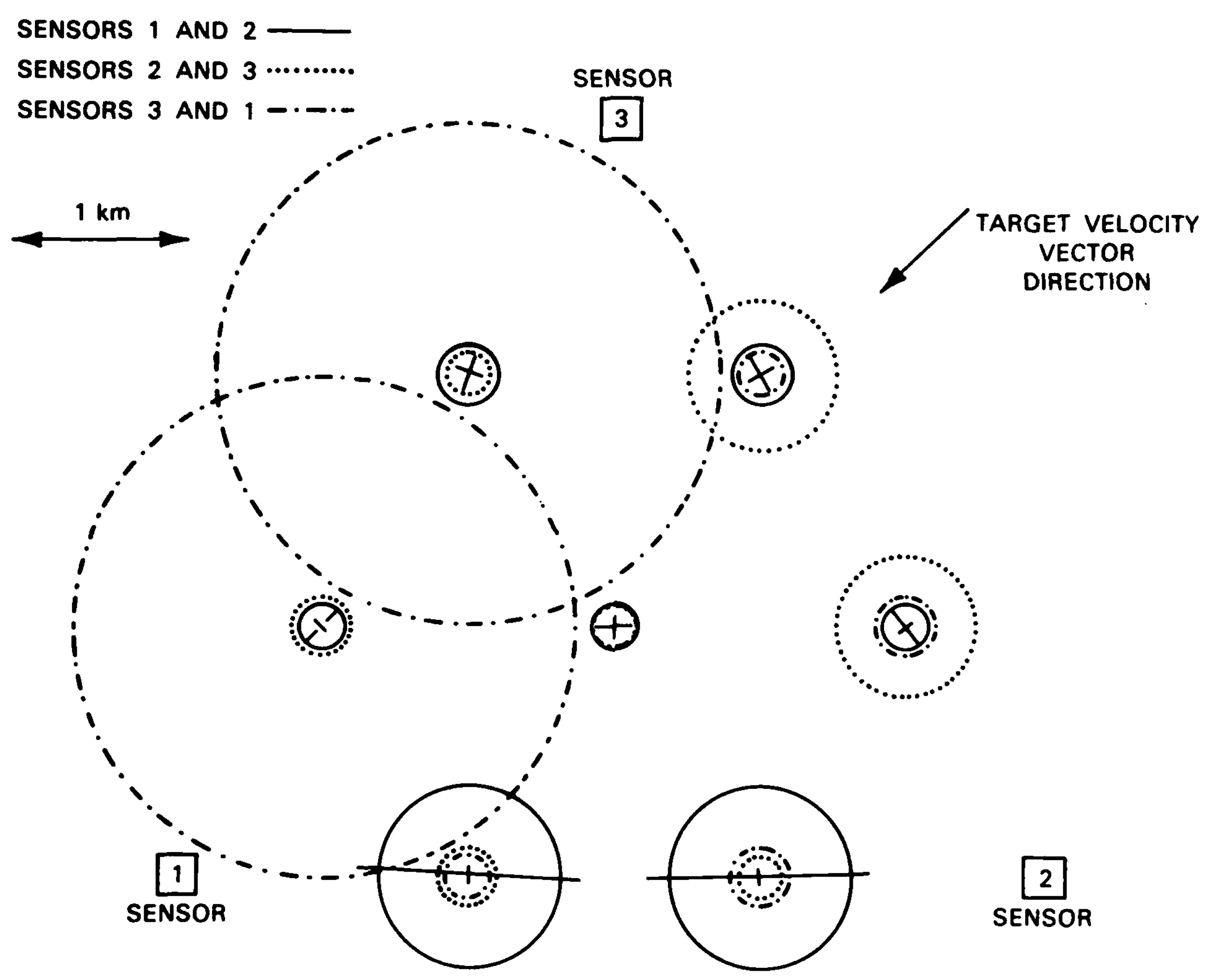


Fig. 2-4. Target Location Accuracies For Possible Position Method, Target Speed = 0.1 M.

128166-N

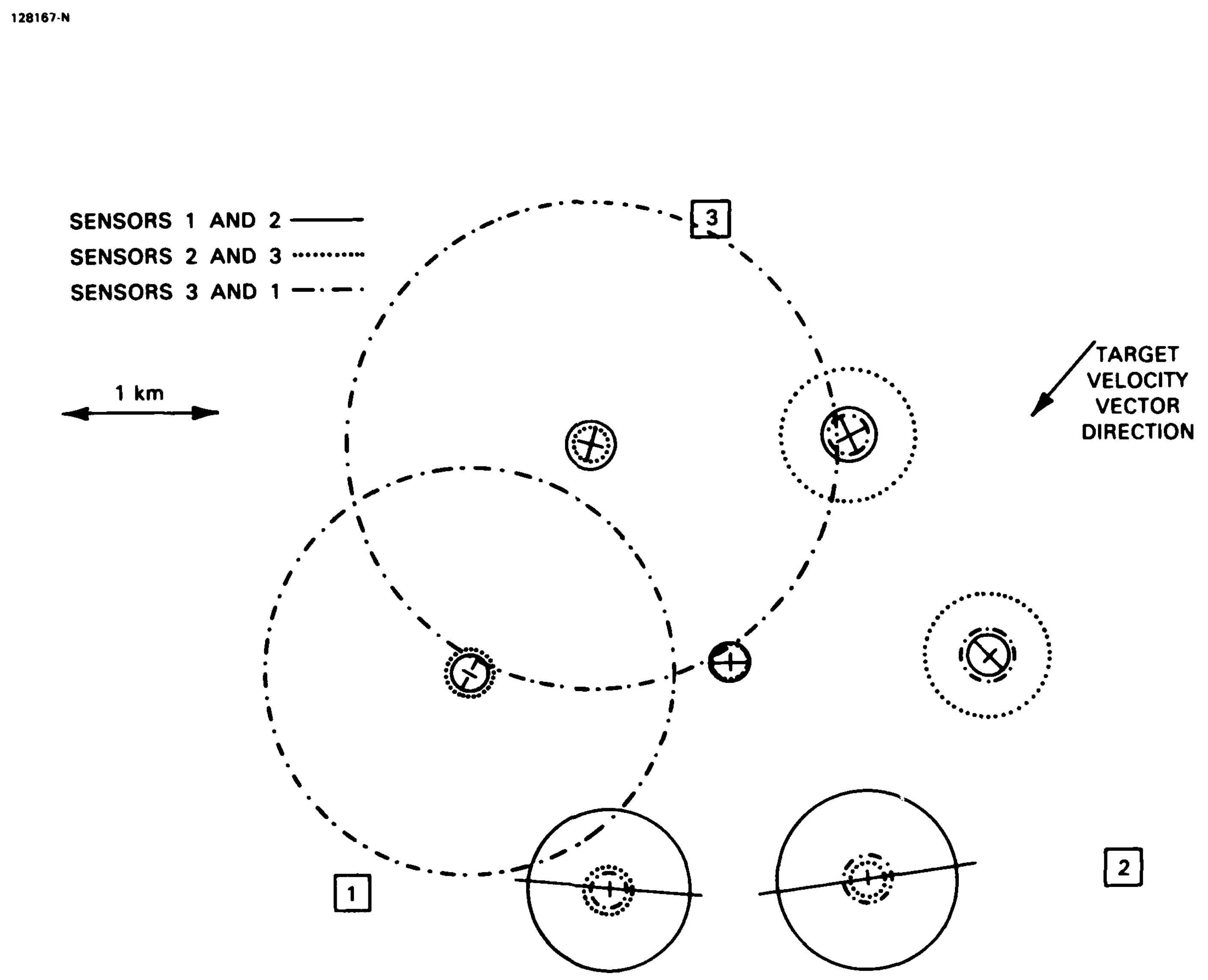


Fig. 2-5. Target Location Accuracies For Reflection Method, Target Speed = 0.1 M.

in Figure 2-4. The figures also show that there can be considerable variation in accuracy from one target location to another for a given sensor pair and from one sensor pair to another for a given target location. A third point made by the figures is that location accuracy ellipses are sometimes far from circular. In fact, the ellipse is highly elongated in all of the situations illustrated with large accuracy CEPs; this fact should be kept in mind when interpreting the accuracy CEPs in these and other figures.

Figures 2-6, 2-7, 2-8 and 2-9 parallel Figures 2-4 and 2-5. The difference is in the target velocity, which is 0.6 Mach in Figure 2-6 and 2-7 and 0.9 Mach in Figures 2-8 and 2-9. As the target velocity increases, differences in accuracy appear in the target locations produced by the two methods. While neither method produces locations which are uniformly more accurate than the other, a brief investigation of various sensor and target geometries indicates that the possible position method produces large location inaccuracies less frequently.

As was mentioned before, the reflection method estimates not only target location but also the time that the target was at the estimated location. Figure 2-10 gives the standard deviations (in seconds) of the temporal 'location' accuracy of the reflection method for the sensor and target geometries of Figure 2-5, 2-7, and 2-9. As one can see, the time accuracy is poor when the location accuracy is poor. Note that the time accuracy can sometimes be very large. A 10 second time error for a high speed target could lead to very large tracking errors.

128168 N

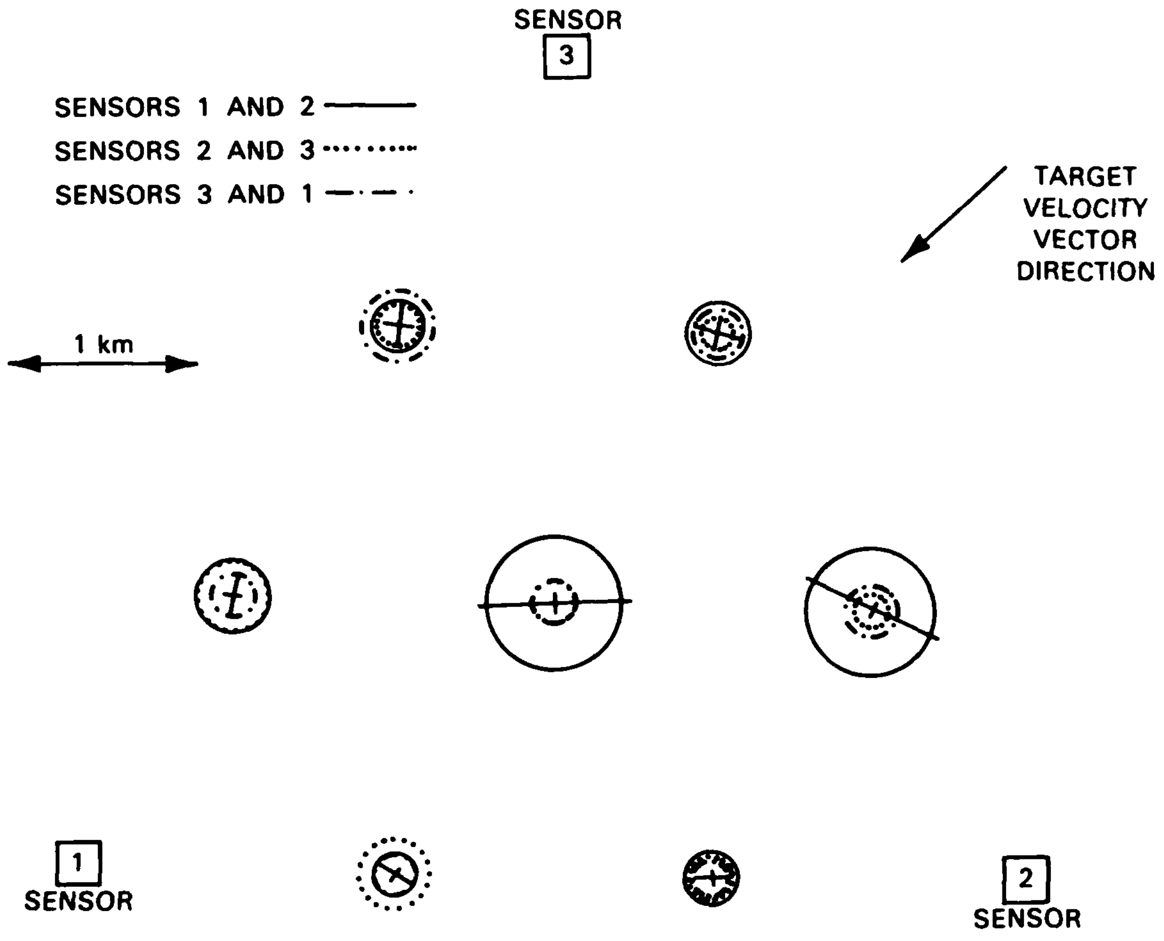


Fig. 2-6. Target Location Accuracies For Possible Position Method, Target Speed = 0.6 M.

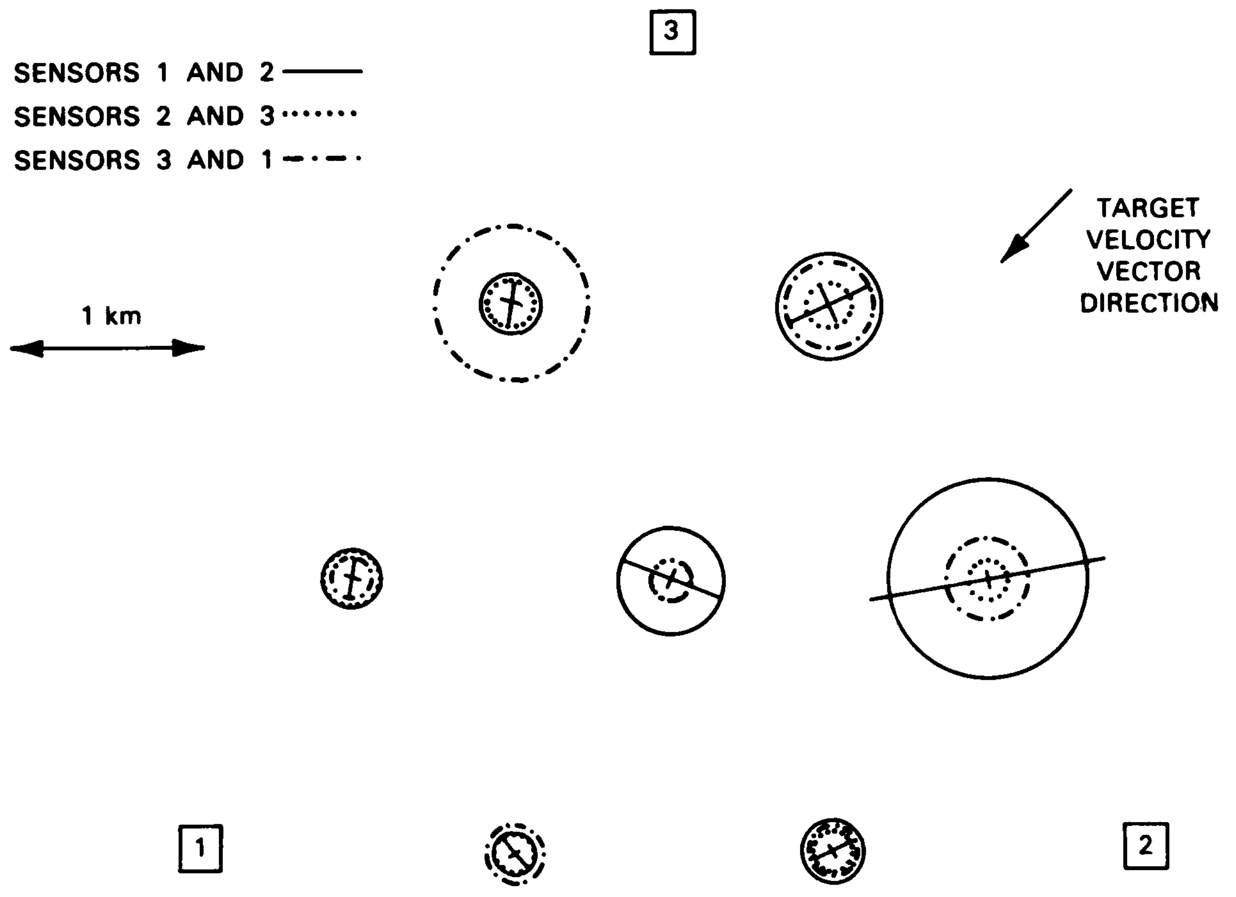


Fig. 2-7. Target Location Accuracies For Reflection Method, Target Speed = 0.6 M.

128169-N

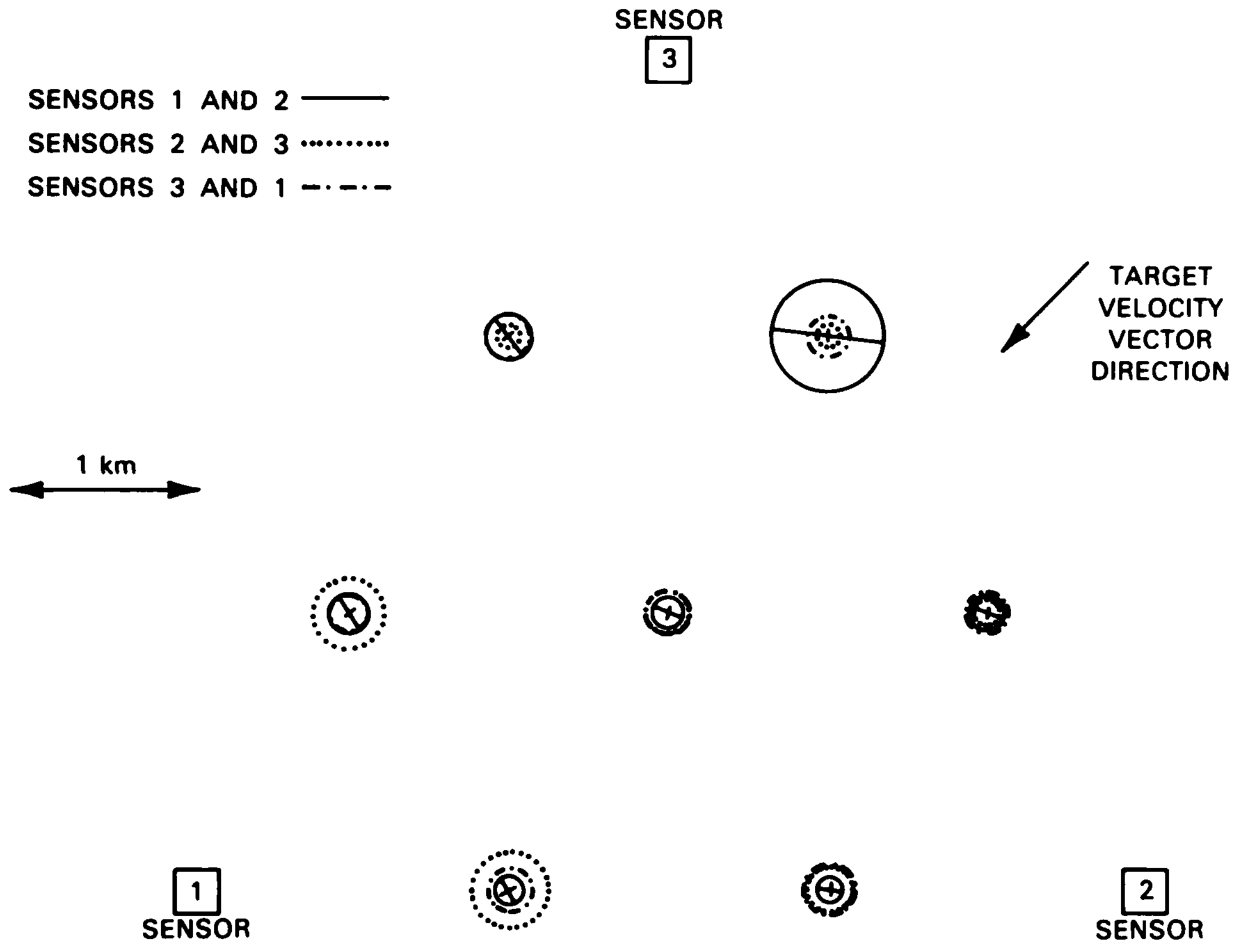


Fig. 2-8. Target Location Accuracies For Possible Position Method, Target Speed = 0.9 M.

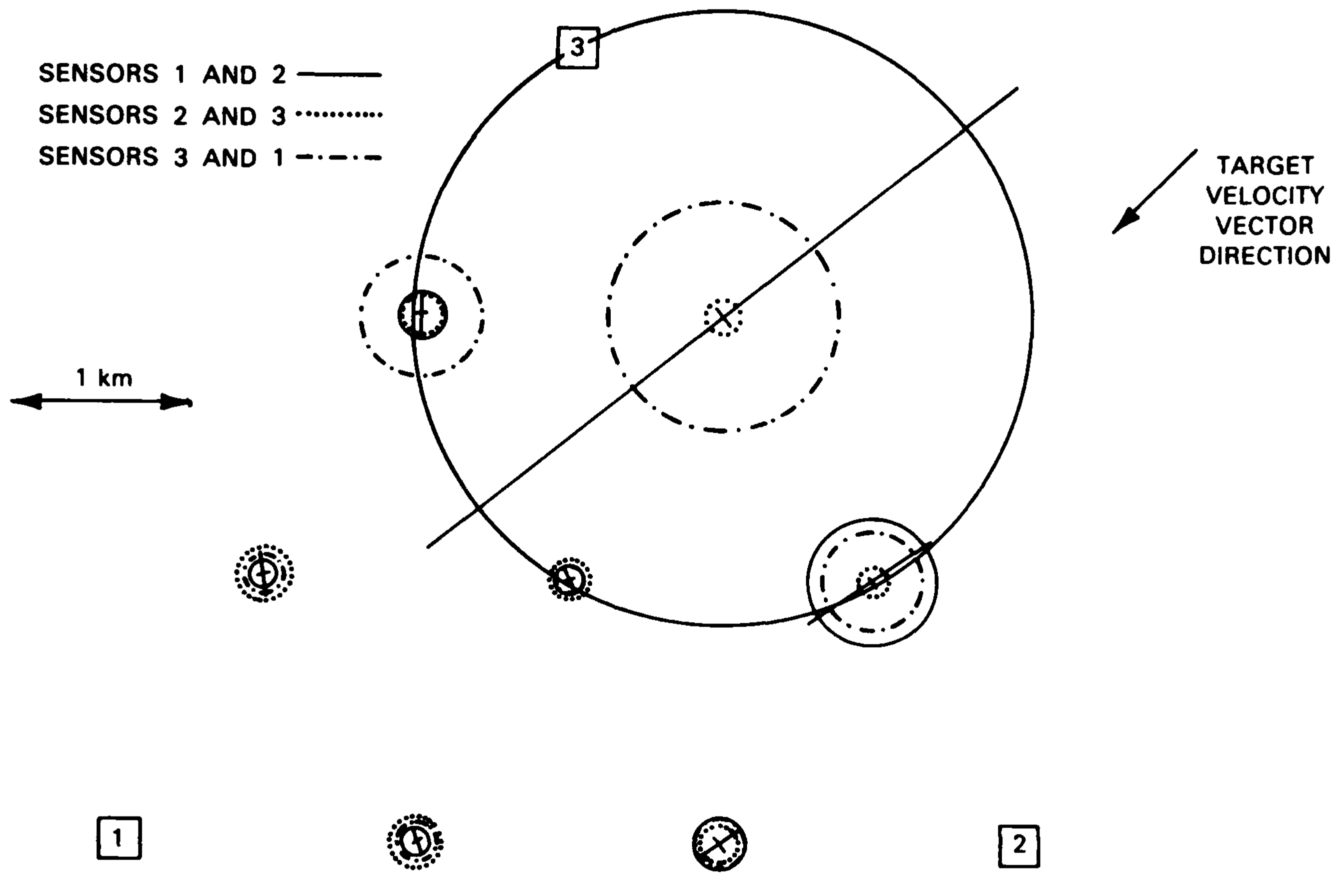
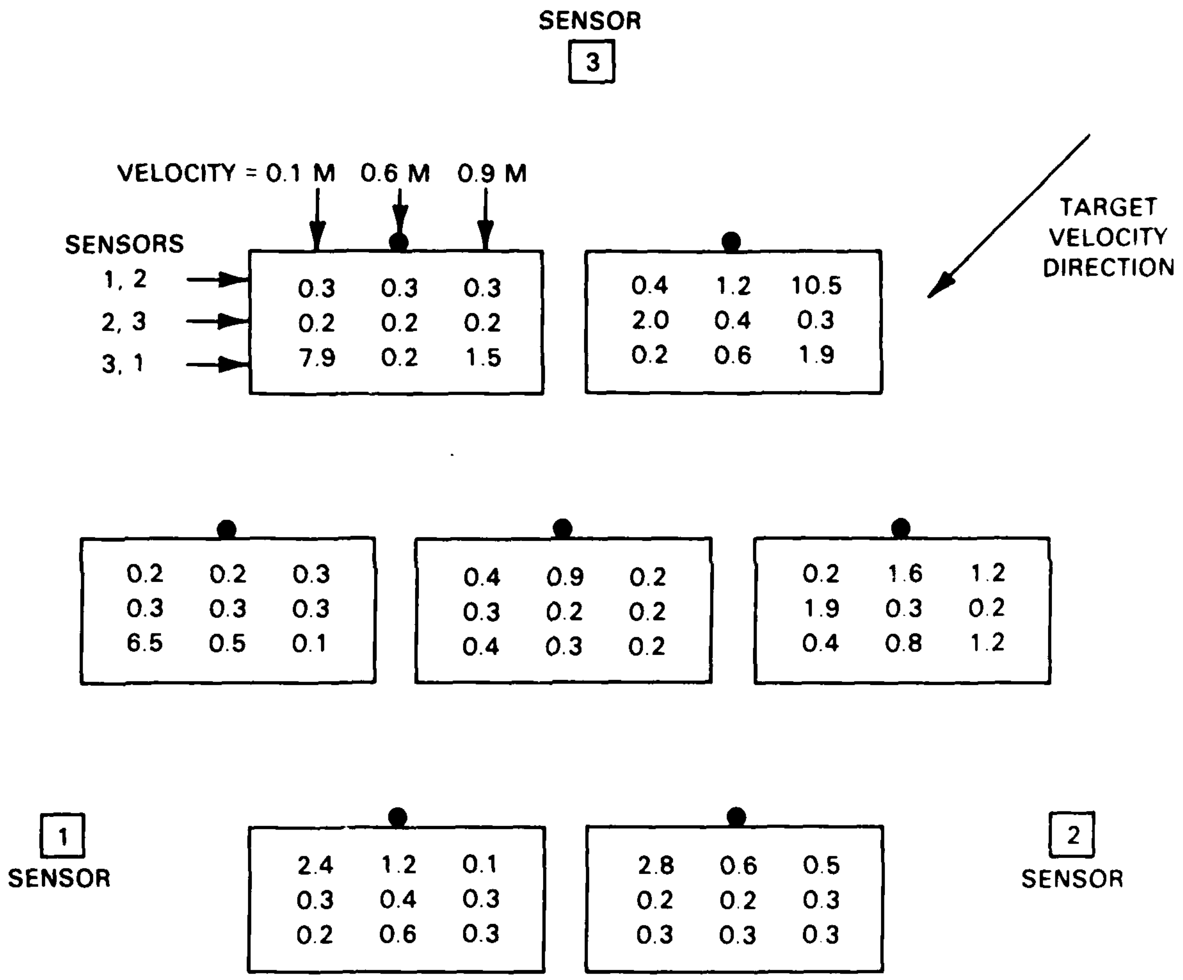


Fig. 2-9. Target Location Accuracies For Reflection Method, Target Speed = 0.9 M.

128171 N



Sensors	0.1 M	0.6 M	0.9 M
1, 2	0.3	0.3	0.3
2, 3	0.2	0.2	0.2
3, 1	7.9	0.2	1.5

0.4	1.2	10.5
2.0	0.4	0.3
0.2	0.6	1.9

0.2	0.2	0.3
0.3	0.3	0.3
6.5	0.5	0.1

0.4	0.9	0.2
0.3	0.2	0.2
0.4	0.3	0.2

0.2	1.6	1.2
1.9	0.3	0.2
0.4	0.8	1.2

2.4	1.2	0.1
0.3	0.4	0.3
0.2	0.6	0.3

2.8	0.6	0.5
0.2	0.2	0.3
0.3	0.3	0.3

128172 N

Fig. 2-10. Accuracy Table for Seven Target Positions Using Reflection Method.

3. ACOUSTIC DATA COLLECTION AND ANALYSIS

3.1 COLLECTION AND ANALYSIS OF MODEL AIRCRAFT DATA

We have considered using remotely piloted model aircraft as experimental tracking targets to provide a relatively inexpensive and convenient way to experiment with a variety of target tracks and different numbers of targets. We recently conducted experiments to collect data on our DSN system for model aircraft and to evaluate this approach for future experimentation. In this section, we describe those experiments as well as our analysis of the collected data. On the basis of this analysis, we have reached two major conclusions. First, although the fundamental spectral peak for these models was within the digital-data collection and analysis range we could not adequately track the targets in the noisy Lincoln environment using our existing hardware and algorithms. The second major conclusion is that there may be better signal to noise ratios at higher frequencies but exploiting that fact would require substantial changes in the test bed system. The general conclusion is that the difficulties of employing model aircraft for DSN experiments outway the potential advantages and that standard aircraft should continue to be used as experimental targets.

The following provides more details of the model aircraft experiments and data analysis.

Three separate experiments were carried out with model aircraft at the main parking lot at Lincoln Laboratory. The two model aircraft used in our experiments were an H-RAY with 0.19 cubic engine and a FLEDGLING with a 0.29 cubic engine. During the initial phase of the experiment, the models were flown separately for several orbits over the parking lot. The second experiment called for the two models, 90 degrees apart, to simultaneously make the same orbit over the parking lot. Finally, the third experiment also had 2 aircraft making the parking lot orbit but this time the aircraft were 180 degrees apart.

The approximate orbit of the aircraft in the various experiments is depicted in Figure 3-1. Also shown in the figure are the location of the three DSN nodes used for the experiments (nodes J, L, and T). On the average, each aircraft took about 40 seconds to complete a single orbit. In all cases both digital and analog recordings were made at each of the three nodes. The analog recording at each site consisted of one channel of data from a single microphone in the array and one channel of voice recording of spotter observations pertaining to the aircraft flight path during the experiments.

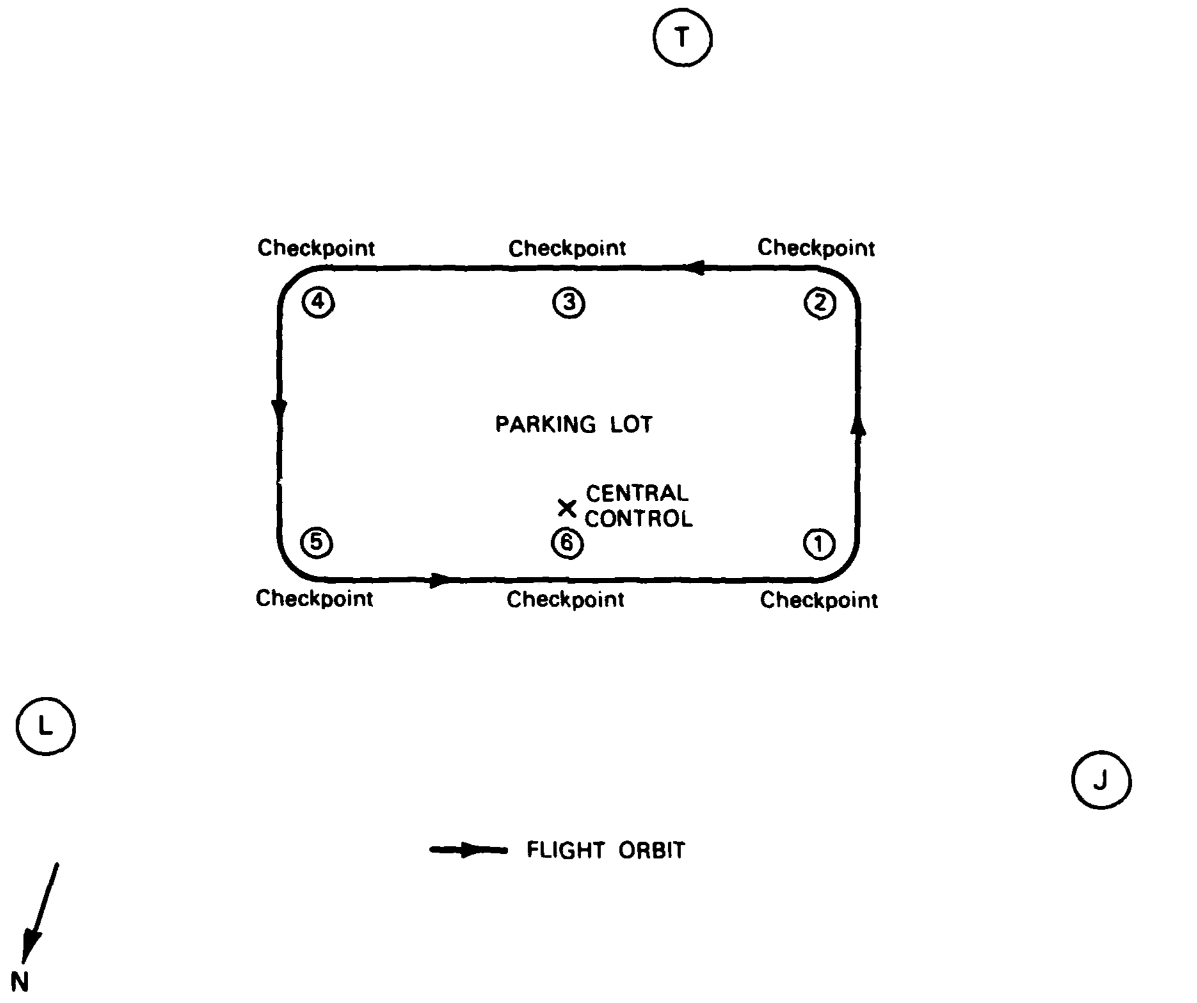


Fig. 3-1. Flight Orbit.

One result of the experiment was the recognition that executing prescribed orbits with model airplanes is very difficult. This complicated subsequent data analysis and interpretation since only spotter observations were available to provide target track information for the quickly changing target configurations. Finally, it should be noted that although the experiments were conducted on a Saturday morning, several other aircraft passed near the area during the experiments. These aircraft generally appeared louder than the model aircraft of the test.

Initial analysis of the data indicated that there would be considerable difficulty interpreting even the single model experimental data. Therefore the data analysis, summarized below, was largely limited to that case.

The analog recordings from the various arrays were reviewed along with the spotter observations. The least noisy recordings appeared to be from node L. For that site, the model aircraft was distinctly audible for at least 75% of the time, the loudest received signal being at the closest point of approach (CPA). At the other two nodes, the plane was audible for a lesser amount of time primarily because of noise from other sources.

Subsequent spectral analysis of the analog data from node L showed a harmonic pattern of spectral peaks with a fundamental at 160 Hz when the model was near CPA. Higher harmonics were present but outside of our normal analysis band due to the effect of temporal anti-aliasing filters with a cutoff of about 300 Hz and to spatial aliasing of our arrays for frequencies above about 200 Hz. At the furthest range from the node the 160 Hz signal from the model was not visible in the spectrum. In all cases other spectral lines corresponding to other aircraft traffic in the area were observed in the data. These lines were often stronger than those corresponding to the model.

Digital data from the acoustic array at node L was analyzed using our standard signal processing techniques to detect targets and estimate the direction of sound arrival. The analysis was restricted to the frequency range 40-180 Hz. The analysis was performed for eight frequencies in each 2 second block of data, for eight elevations in the 0-80 degree range and for 120 azimuth steps. The model aircraft signal to noise ratio was small and this analysis confirmed that successful tracking would not be achieved without changing the experimental site to a less noisy one and changing the hardware and software to handle higher frequencies.

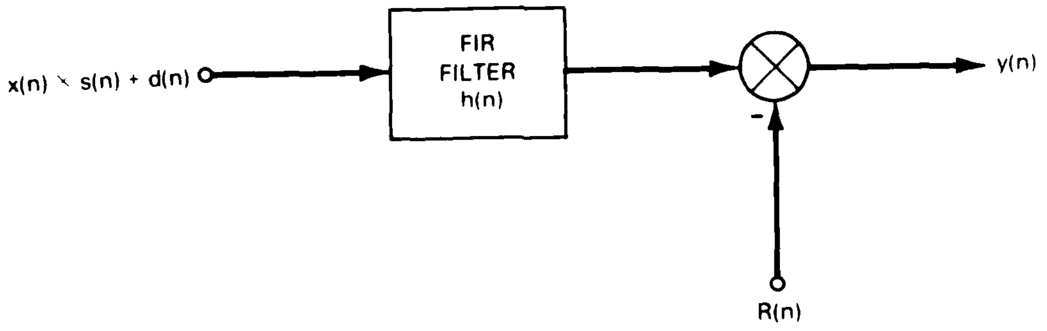
Separate static acoustic measurements have also been obtained and spectra have been measured for other engines that might be used in model aircraft or remotely piloted vehicle (RPVs) experiments in order to determine whether slightly different engines might change the general conclusions noted above. One of these was a SAITO FA-40 4 stroke engine and the other was the engine for a XQBM-1-6 RPV made by Centro Corp., Dayton, Ohio. As expected from knowledge of the engines and the physics of noise generation both of these had lower fundamental frequencies than the engines used in the flight tests described earlier. The fundamental for the SAITO was about 90 Hz and about 80 Hz for the XQBM, for typical operational RPMs. Although this would be of some help, it appears that their noise levels and the difficulty of executing experiments still favor using full size aircraft for experiments in the foreseeable future.

3.2 Frequency Analysis of Single Channel Acoustic Data

Frequency analysis of individual data channels collected at a DSN node is an important part of the nodal signal processing. The spectral characterization of the signals is of interest for two purposes. One is to select frequencies corresponding to the location of narrowband energy peaks from targets. Spatial analysis then can be performed only at those frequencies. Secondly, a more detailed description of the various harmonic components can be used to classify and differentiate between targets. In this section we report on some investigations of alternatives to the periodogram methods now in use in the test-bed nodes. The frequency analysis method currently used in the test bed finds the N highest spectral values in the periodogram of a section of data.

We have identified four alternative approaches for investigating the size and location of spectral peaks in single channel data. One of these, adaptive line enhancement, has been tried on synthetic data. The adaptive line enhancement approach and preliminary experimental results are detailed below. A second possibility is to build upon pitch detection techniques developed for speech processing. Extensive research has been done in the pitch detection area which parallels our problem in that the period of a signal (voiced speech) is estimated in the presence of broadband noise. The third approach is to consider the problem as that of detecting a known signal in additive broadband noise. This is a classical detection theory problem which has received considerable attention (refer to Reference [3]). This known signal approach is practical when particular types of targets are of interest and the system has *a priori* knowledge of the source characteristics. The fourth approach is a generalization of the third. In this approach it is assumed that the signal to be detected has been delay-modulated with an unknown slowly time-varying function. A detection strategy for this situation has recently been proposed (see Reference [4]). The assumption of slowly time-varying delay-modulation in the periodic signal is a characteristic which has been observed in much of the DSN data we have collected.

The adapative line enhancement approach that we have investigated is derived from an adaptive filtering formulation for noise cancellation by Widrow (see Reference [5]). Given a signal x(n) = s(n) + d(n) and a reference signal r(n) that is strongly correlated with d(n), Widrow proposed a strategy for separating the signal s(n) from the noise d(n). This algorithm is summarized in Figure 3-2. The basic design problem is to find an FIR filter h(n) of a given order so that the energy in y(n) is minimized. Widrow showed that the resulting y(n) can be treated as an estimate for s(n). In the application of this idea to line enhancement, we use r(n) = x(n-m) for some delay m. Furthermore, s(n) is assumed narrowband and d(n) broadband. Using the assumption that d(n-m) is not correlated significantly with x(n), it is argued that minimizing the power in y(n) leads to an FIR filter h(n) that seeks to minimize d(n) while maintaining the power in s(n). However, this argument also requires that the filter order be large enough to ensure that the filter response to s(n) can be made to correlate strongly with s(n-m) while minimizing its reponse to d(n). With this requirement, the filter h(n) will be a comb filter with its spectral peaks at the frequency locations of the narrowband signal s(n). This filter can then be used as a replacement for the spectrogram in our previous approach to the frequency location problem.



128174-N

Fig. 3-2. Adaptive Line Enhancement Filter Model.

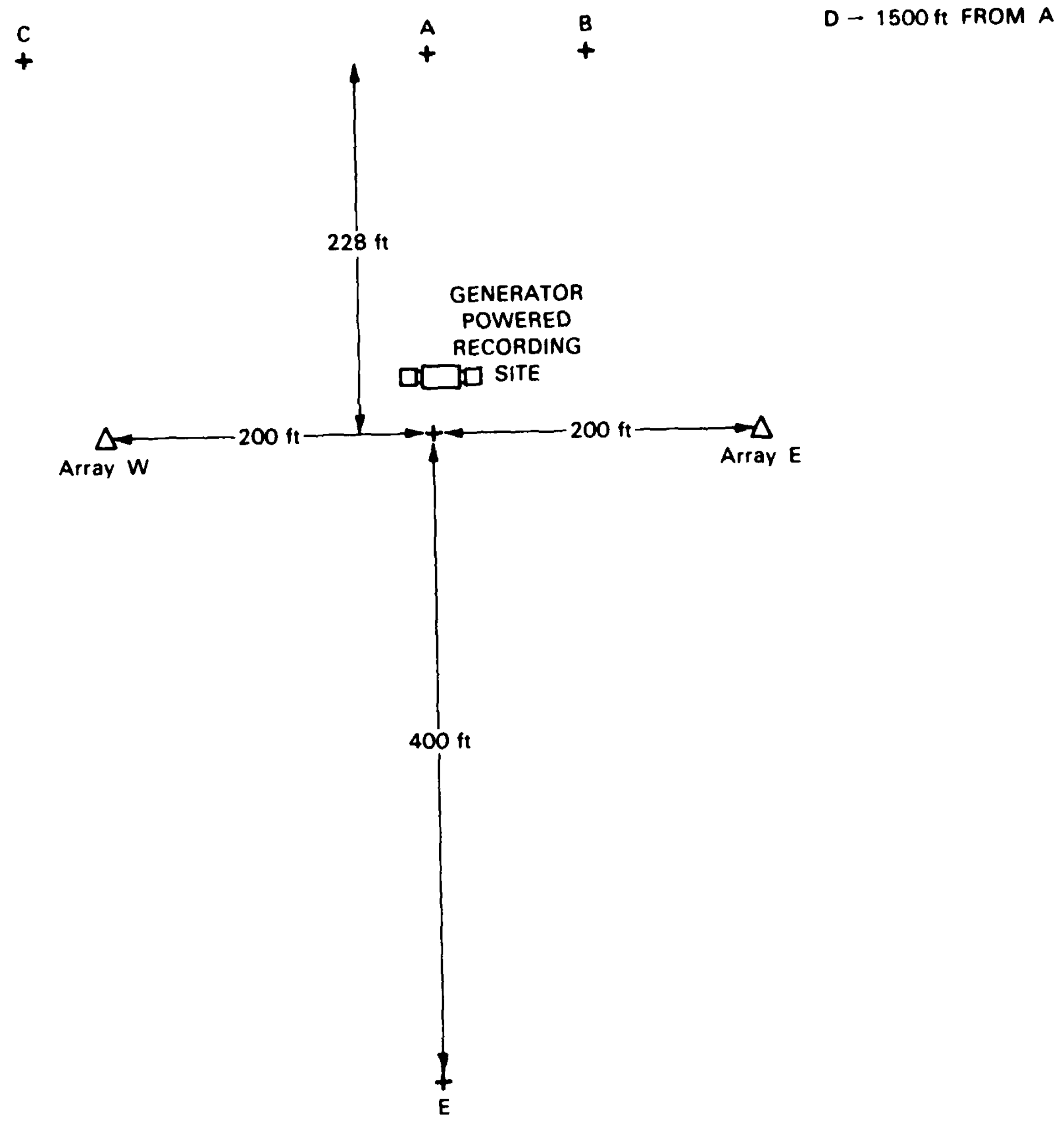
To determine the practical feasibility of the above approach, we experimented with simulated data. In these experiments, the signal s(n) was taken as a linear combination of 1, 2 or 3 cosines of various amplitudes and periods. The noise d(n) was simulated as white noise and the signal to noise ratio ranged between +10 and -10 dB. The delay m was chosen to be 200, much larger than the periods of the cosines. Various filter orders were tried for the FIR filter h(n), ranging from 20 to 150. Preliminary results show that the technique has not provided better performance than straightforward spectrogram analysis except under very specialized conditions. Further investigation will be required to evaluate more fully the adaptive line enhancer approach to harmonic detection and measurement.

3.3 Validation of Mobile Node for Remote Data Collection

In late September, mobile node one was shipped by flatbed truck to Salt Lake City, Utah for an acoustic data acquisition experiment. This was done in conjunction with a radar measurement experiment being conducted for DARPA and constituted an excellent opportunity to subject the DSN mobile node to realistic transportation and remote operational conditions. The radar experiment involved a radar mounted on a Bell L206 Long Ranger II helicopter that was also used for DSN acoustic measurements. This test involved the use of the acoustically quieted generator for power and field testing of the latest data acquisition program, SOUND, which stamps the data with satellite time accurate to within about 1 ms relative to Greenwich Mean Time. It represented the first truly remote operation of the DSN test bed.

The experimental layout is depicted in Figure 3-3. Twelve microphones were deployed in two six-element arrays marked W and E. The arrays were of two different designs as shown in Figure 3-4. Array W was a standard DSN array composed of two concentric equilateral triangles. Array W was derived by Haubrich (refer to Reference [6]) as a very desirable array because the samples in lag space generated by the array are uniformly distributed out to the maximum lag. This design is intended to maximize the resolution for a given minimum spacing. Both arrays have an upper aliasing limit of 227 hertz.

The acoustic measurements of the helicopter consisted of several minutes of stationary hovering over A and B, continuous straight line runs from C to D at 7.6 meters/sec and some recordings of the helicopter on the ground at point E. The nominal altitude of the helicopter was 33.5 meters. Recordings of the ambient noise, including the generator, were made between helicopter runs.



128175-N

Fig. 3-3. Salt Lake City Experimental Layout.

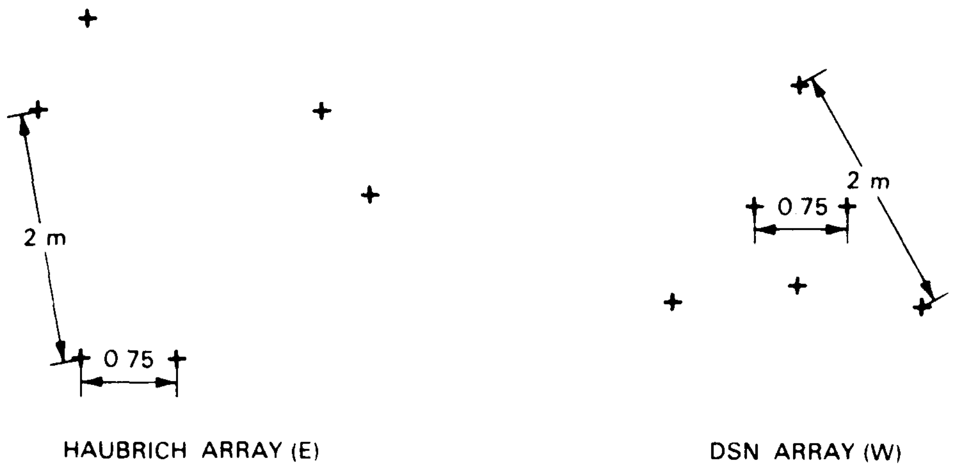


Fig. 3-4. Array Designs.

Typical spectra from both arrays for the helicopter hovering over station A are shown in Figure 3-5. The helicopter's main axis was parallel to the line joining the two arrays with the tail rotor closest to the E array. The fundamental frequency of the main rotor and tail rotor was 13 hertz and 83 hertz respectively. The fundamental and first two harmonics of the main rotor were 3 dB stronger when observed from the rear of the helicopter (array E). The higher harmonics and the tail rotor components were stronger by the same degree in the forward direction. Similar changes in the distribution of power in the harmonics as a function of orientation have also been observed for a moving UH-1 helicopter as reported in a previous SATS (note Reference [7]). We hope to eventually exploit such knowledge about targets and data characteristics in actually performing target classification and tracking functions.

A typical spectrum from the ambient noise recordings superimposed on a helicopter spectrum is shown in Figure 3-6. The generator component at 60 hertz is not visible. The general shape of the ambient spectrum is consistent with wind noise as the major source. The level is 67 dB SPL at 13 hertz and 59 dB SPL at 83 hertz. The signal to noise ratios at the two major fundamentals are 21 dB for the main rotor and 29 dB for the tail rotor.

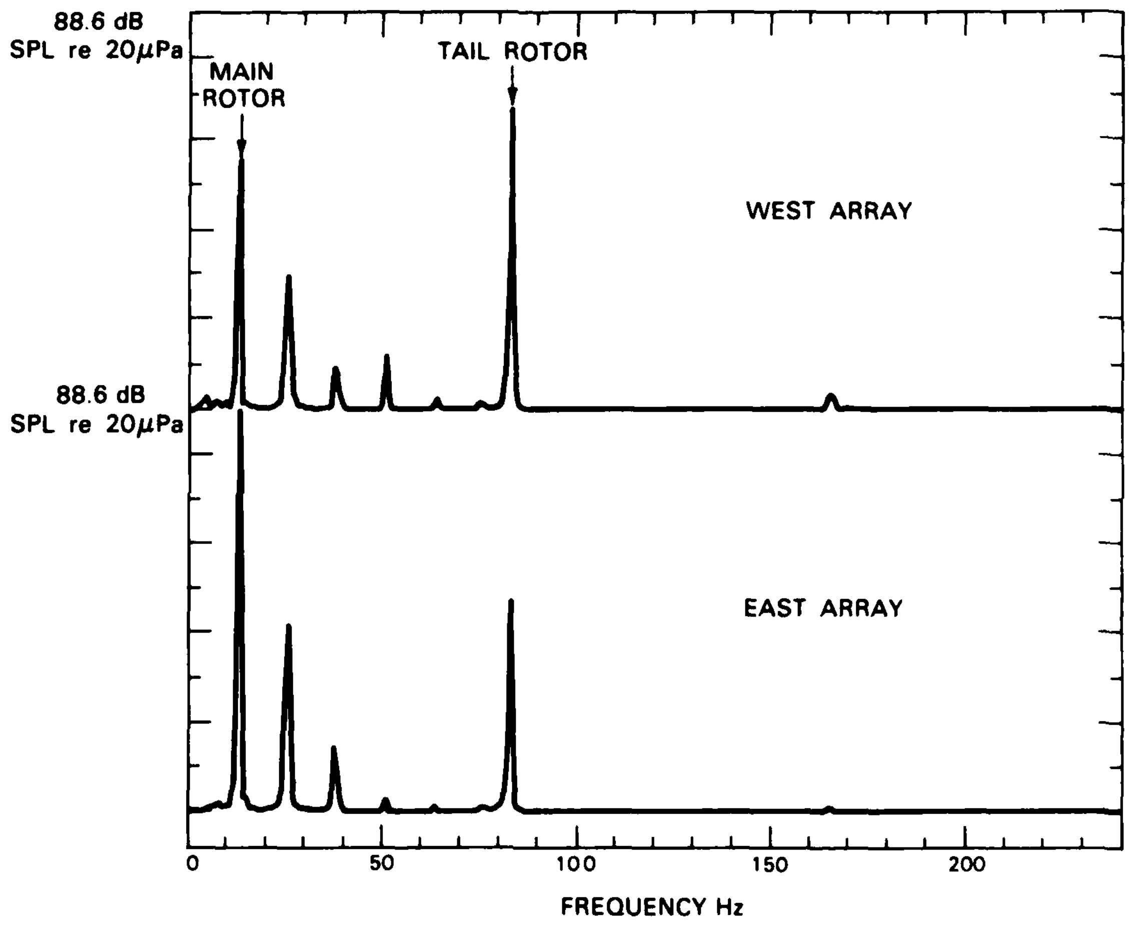


Fig. 3-5. Power Spectra Helicopter Hovering A.

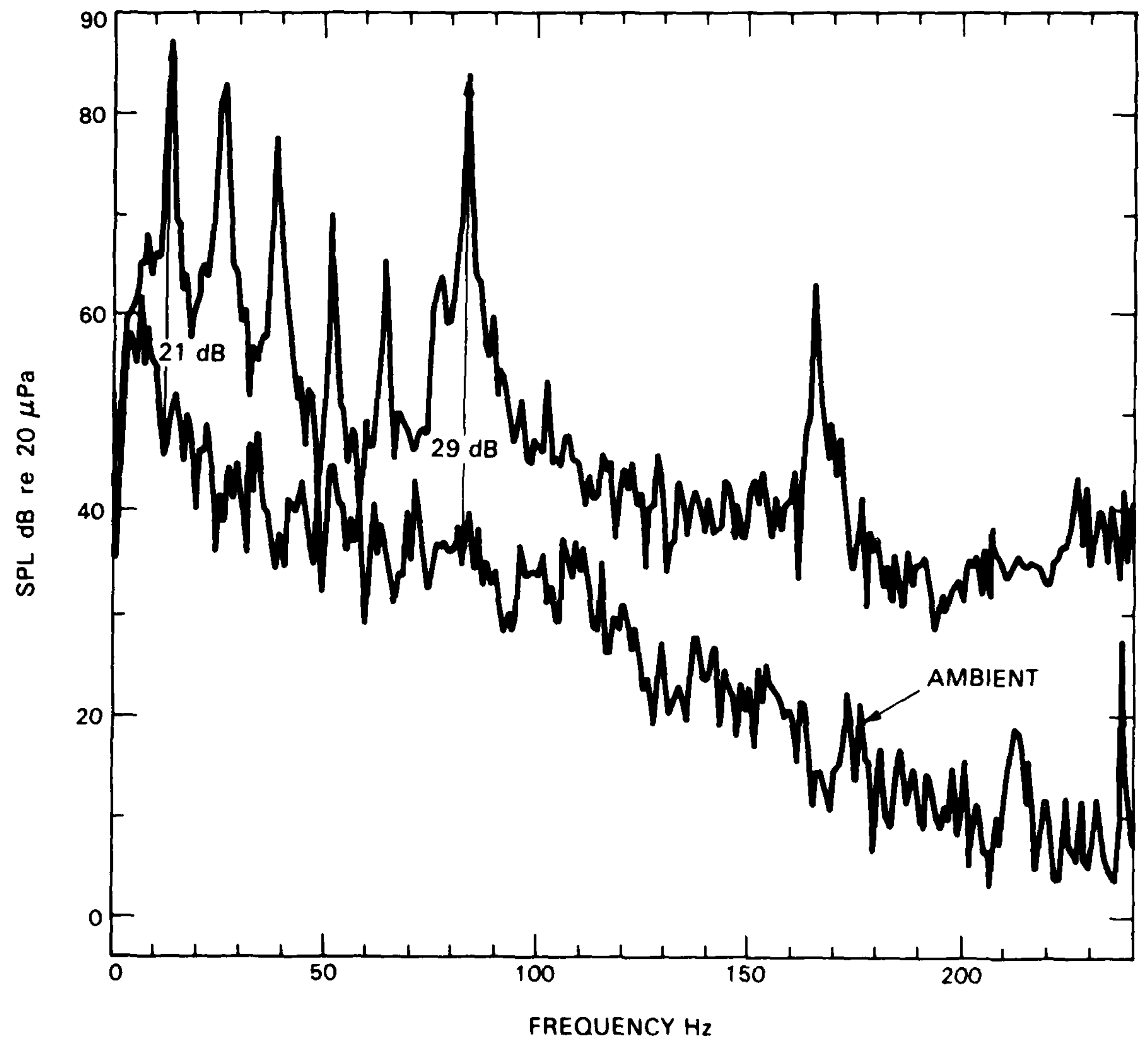


Fig. 3-6. Ambient Noise Power Spectrum.

128178 N

Frequency-wavenumber analysis of the data using the maximum likelihood method was performed at the frequencies of the major peaks in the spectra. A typical analysis result obtained using the helicopter tail rotor frequency of 83 Hz showed the two arrays providing consistent elevation estimates of 20 and 26 degrees and providing a location estimate to within 5 meters of the true location. This location accuracy is about 7% of range for the case considered.

The resolution of the Haubrich array (E) was better than the array W. Array W's resolution could be increased by increasing the size of the outer triangle, but this would be at the expense of the beampattern. Further analysis and usage of the Haubrich style arrays is anticipated.

With the exception of the failure of anti-aliasing filters on two of the acoustic channels, the DSN test bed performed without failure throughout the five days of setup and testing at Salt Lake City. The success of the remote operation, time stamping and recording under generator power confirmed the utility of the DSN node as a mobile data acquisition system.

4. TEST BED SOFTWARE

During the past six months considerable progress has been made in the design and implementation of software necessary to support cooperative tracking experiments in the test bed with internodal message exchange through land lines. This software is primarily for the Motorola 68000 Versamodule systems that are installed in three test bed nodes and for the PDP 11/70 computer that is serving as a communication switch and user interface computer. This development of test bed capabilities has involved:

- Development of nodal software to support multiple processes
- Development of nodal software for interprocess message communication
- Development of mechanisms for debugging programs running remotely in the nodes
- Implementation of a user control interface to the network
- Development of software to support internodal message switching through the PDP 11/70 computer
- Conversion of azimuth and position tracking programs to run in the nodes under the message based operating system

We have used the C language and a Unix-like form for all 68000 software. This has facilitated the transfer of software from standard Unix systems to the test bed nodes. Because of this, a great deal of software development has taken place in the user friendly Unix environment. As a direct result of this, Unix programmers are able to produce nodal software with minimal learning time.

In general we have attempted to implement the minimal features needed to support applications. Additional features have been added only as the need became apparent. This development process has resulted in the recognition of a number of desirable changes that should be made as the test bed evolves. Both the present system and a discussion of some of these desirable changes are described below.

4.1 Tasks and Task Scheduling

The multi-tasking facilities in the DSN run-time system provide several rudimentary task synchronization facilities and a simple task scheduler. The following is a description of some of the mechanisms involved in these concurrency features.

Nodal software is organized into a set of concurrent tasks. At any given time, one of these tasks is actually being run by the processor and is said to be active. The remaining tasks are inactive. An inactive task may be either ready, waiting, or terminated.

A task may wait for one of three events. It may wait for the value of a flag (i.e., an integer variable) to change; it may wait for a specified time to elapse; or it may wait for another task to terminate. Run-time system calls are provided which allow a task to wait for an event. Additional system calls permit a task to terminate itself or to give up the processor, (i.e., to become inactive while remaining ready). The currently active task becomes inactive only by executing one of these system calls. A task which is ready may be made active by the scheduler any time the currently active task becomes inactive. A task which is waiting becomes ready after the event for which it is waiting occurs. A task which is terminated remains so until the load module is restarted.

The scheduler runs as a task that becomes active whenever any other tasks becomes inactive. It maintains a list of ready tasks and a list of waiting tasks. Each list is maintained as a first-in-first-out queue. When the scheduler runs, it performs three basic steps. First, it moves the previously active task to the end of the appropriate list. Second, it determines which of these events that the tasks are waiting on have occurred and it makes the appropriate task(s) ready by moving them from the wait-list to the ready-list. Finally, it relinquishes control of the processor to the task at the head of the ready list.

We have found this scheme generally to be workable but some limitations have become apparent as the size of the system has grown. One shortcoming is that there is inadequate protection of the supervisor from the user tasks and of different user tasks from each other. In addition the almost complete lack of system-level error handling makes run-time error recovery difficult. At a more detailed implementation level a number of efficiency shortcomings have been identified along with ways to correct the situation. Finally, the notion of task cooperation in which the tasks themselves are responsible for relinquishing the processor has resulted in a very simple scheduling system. However it complicates the user tasks and when application tasks change it is sometimes necessary to change when and how they relinquish control. This is inconvenient in the present system and could become unacceptable in large systems. We have begun to refine the system to correct some of these problems.

4.2 Message Communication

Test bed communication software has been developed to support reliable point-to-point message communication and the simulation of radio broadcast services using land lines. The system consists of the following elements:

- A utility package to format messages
- A single processor interprocess communication mechanism that supports multiple message streams between processes
- An interprocessor communication layer that supports both point-to-point and broadcast communication services between processors

Application programs call the message utility package to format messages. After formatting, messages are passed from this application layer to the interprocess communication layer which is responsible for queueing them up ready for delivery by the interprocessor communication layer.

The basic interprocess communication layer, IPC, was designed and developed by CMU. We have now modified it to meet the specific requirements of the DSN test bed. It supports separate message streams using a first-in-first-out queue structure for each stream. These message streams have user assigned names associated with them and are called message ports. IPC messages are written and read on named ports as shown in Figure 4-1. IPC provides for allocation of named message ports, queueing of messages on ports, reception of messages from ports and message garbage collection. Messages can be received from multiple processes on a single input port.

The interprocessor layer of communications software was built on top of the modified CMU package. It is implemented as a router task running in each processor. The router tasks provide message routing to the destination address, broadcast service, flow control and error handling. Router tasks monitor messages on incoming serial lines and local ports. Message routing is based upon the destination address and message class specified in the message header. Message headers contain destination address and message class information. Based on this information the router tasks send messages out on serial lines or queue them on local IPC ports as shown in Figure 4-2.

The present software is an interim solution to interprocessor communication in the test bed. The longer term plan is to replace the router task layer with the DOD standard TCP/IP. The TCP/IP protocol will support both the serial line and radio environments. TCP will contain all facilities to support reliable point-to-point service and IP will support the broadcast service.

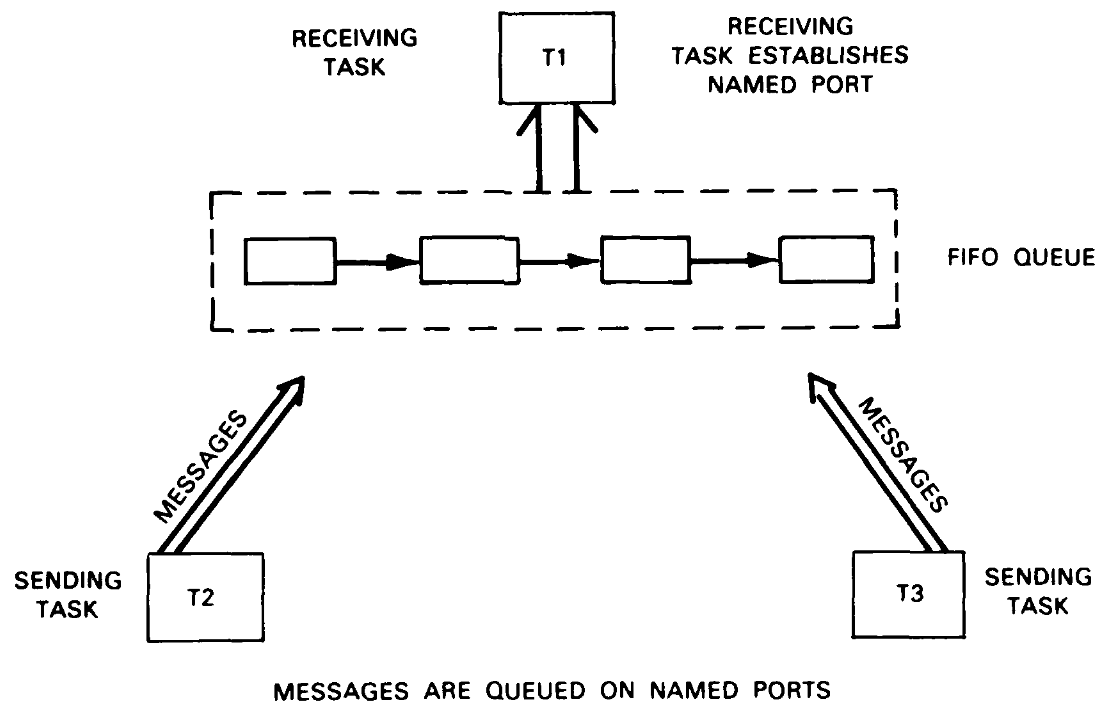


Fig. 4-1. Interprocess Message Passing.

128179 N

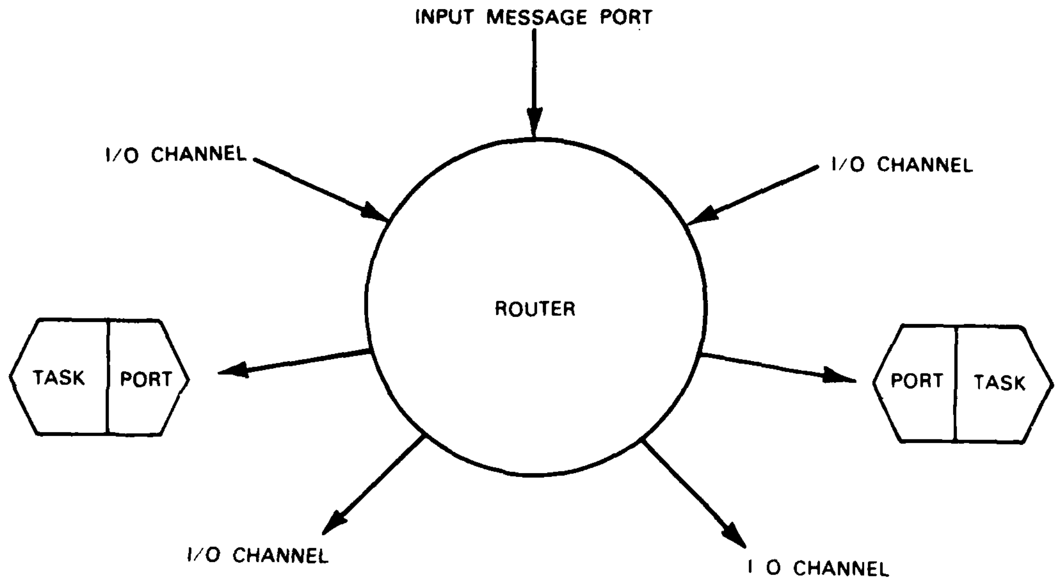


Fig. 4-2. Router Task.

4.3 Debugging Facilities

Debugging capabilities in the run-time system are provided via a set of system routines. These debugging routines are called by a task named Exec, which executes in each test bed node. Users interact with Exec by means of a program called Monitor that runs on the 11/70 and exchanges messages with Exec over the test bed network. The user interface to Monitor/Exec is described in Section 4.4. Let us begin by describing the implementation of the operating system debugging routines and their incorporation into the Exec task.

The run-time system debugging routines fulfill two basic requirements. They provide debugging functions (such as breakpointing), as well as the ability to determine the status of a task or to examine and modify memory. Further, such debugging is task-specific; it is possible to debug one task without affecting the performance of other tasks in the system. Task specific debugging was provided by adding the capability of task suspension to the scheduler.

Suspension is caused by one of three conditions. A task will be suspended if it hits a breakpoint; after it executes a single instruction in trace mode; or by a 'suspend' system call. In all cases, a suspended task is de-suspended, or resumed, by an invocation of a 'resume' system call.

Several debugging-related system calls were implemented in addition to suspend and resume. One pair of calls provides for the setting and clearing of breakpoints. Another pair permits the initiation and termination of trace mode. In the trace mode, a task is suspended after the execution of each instruction. (It is, conceptually, the same as setting a breakpoint after every instruction.) Finally, a set of calls allows the examination and modification of memory.

These debugging-related calls were used to implement the Exec task. The Exec accepts user command and executes them. In the case of debugging commands, it will convert a command into a sequence of one or more system calls. For example, the command "break joe 2500" is the command to set a breakpoint for the task joe after the instruction at address 2500. The Exec checks to see that there is a task called joe and that the breakpoint 2500 has not been previously set; if both of these conditions are met, then it will set the breakpoint.

We have found this approach to be quite satisfactory. However, two primary problems remain.

In theory, the Exec could accept its commands from any input source, e.g., either a serial I/O port or a message port. However, it was written initially to accept commands only from a message port. This initial condition was for an carly version of the message system. When it came time to implement an expanded message communication system, there was considerable difficulty debugging it, since the debugger used the system message communication routines to interact with the user. This caused us to develop a serial-I/O version of the Exec and maintain two different versions of the Exec. We expect to resolve this problem by changing the system to use the Unified I/O interface described in Section 4.6. The Monitor executes on a machine which also has a copy of the load module of the program being debugged. Thus, given a symbolic name, the Monitor could convert the name to an absolute number using the symbol table of the program.

4.4 User Interface and Internodal Broadcast Communication Emulation

As noted above, a Monitor program has been developed that executes on our 11/70 computer and is used to interact with tasks running in remote processors. Monitor takes commands from the user's keyboard, converts these to messages, and sends the messages down to the nodes. It also takes messages from the nodes and displays their contents on the user's terminal screen. The basic capabilities of the Monitor program as a user interface include:

- Controlling and debugging of tasks in a remote processor by interaction with the Exec task in that processor
- Linking to a particular node/cpu/port and having commands or data that a user enters on his keyboard sent to this port
- Reading commands and data from a file
- Spooling azimuth and position track messages to a disk file on the 11/70 computer
- Injecting azimuth and position track messages into the test bed from a file on the 11/70 computer; this is called reverse spooling

There are two improvements now being added to the Monitor program. The first is to provide a split screen for the user so that messages arriving from different nodes do not interfere visually with the input of commands. The second is in providing the capability to direct a command to a node/cpu/port other than the one that the user is currently linked to. This enables a sequence of commands which start processes in separate nodes to be stored in a file and then issued rapidly such that the processes start within a short time of each other.

The Monitor program that runs on the PDP 11/70 has also been designed to provide the emulation of internodal broadcast. This software sends copies of all azimuth tracks received from any test bed node to all the other nodes. A first version of this software has been written and is being refined and debugged.

4.5 Tracking Software Conversion

Tracking algorithms have been converted for use in the Versamodule systems in the test bed nodes. This conversion involved interfacing the algorithms with the message based communication system, converting the floating point operations to fixed point operations and providing a capability to remotely set and examine tracking parameters while tracking is in progress. This last capability will enable experimentation with the effects of dynamic parameter changes.

4.6 Unified I/O

The software interface to the current message based interprocess communication system is distinct from the software interface to the physical I/O calls. It now appears desirable to unify the I/O and message passing into one set of Unix-like calls. With this approach, all task I/O can be done by reads or writes, which may go to physical I/O channels or to message ports depending on the device name opened.

The advantage of this approach is that procedures may be written and tested without regard to whether they will be run with a local terminal as control device, or with a remote terminal over a circuit or a message switched network. This would simplify coding and permit software to be developed on a general purpose Unix system and then be integrated into nodes with little or no alteration. The basic features of the proposed Unified I/O scheme are:

- Tasks use a Unix-like open( ) system call to create named ports on which to send or receive messages
- Tasks read messages from a port by using one of the standard Unix forms of read( )
- Tasks write messages using one of the standard Unix forms of write( )
- Physical I O ports are differentiated from message ports only at creation

Two additional extensions to the basic scheme are also being considered. Under Unix a read or a write will not return to the caller until it is completed. In a DSN nodal environment there is the need to continuously check on a number of physical devices and ports to see whether any data or messages are available to process. In the existing version of the software we have implemented immediate return routines that allow for this to be done. One extension will be to add a user specified timeout to read operations. If the request cannot be fulfilled within the timeout, then a return will be made to the user with a special return code. This will constitute a generalization of the immediate return feature. Another extension will be to provide for reply ports as part of the basic message system. In a conventional Unix environment scanf() and printf() are used as the mechanism for an interactive dialogue between the program and its user. Reply ports as part of the basic message system can be used to provide a similar capability in a remote message base environment.

5. TEST BED HARDWARE

5.1 Operational Improvements

As noted elsewhere in this report a mobile node (Mobile 1) has successfully completed a field experiment in Utah during this period. Based upon past experience several modifications were made to that node prior to its successful field use. These involved improving the flexibility and convenience of the array deployment, changing the shock isolation of the equipment and making improvements to the ancillary communication equipment that is needed to conduct experiments.

Equipment for the other two mobile nodes has been checked out and installation is underway. The Mobile 1 improvements are being incorporated into those nodes. No other changes have been made to the operational test bed hardware during this period except for routine maintenance and the detection and correction of minor problems.

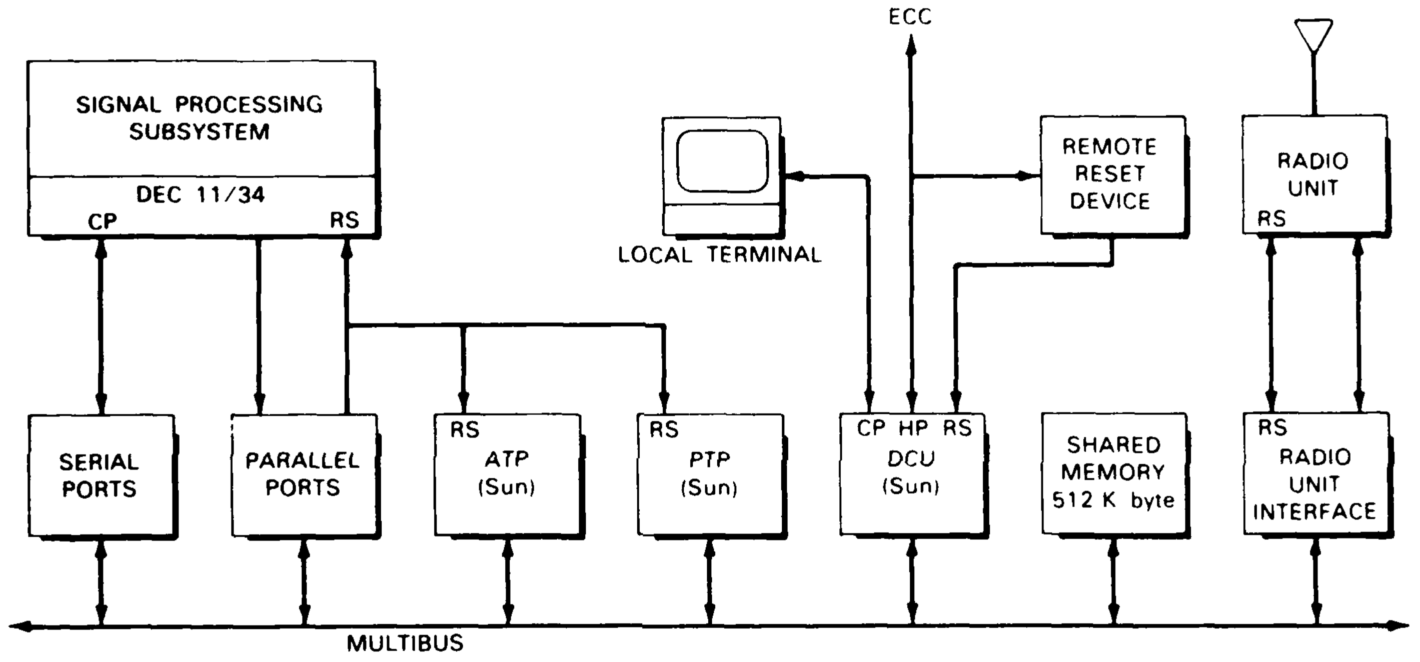
Hardware problems have been reoccurring with the Motorola Versamodule CPU cards that are installed in three test bed nodes. The current plan is to make use of these units only until they can be replaced as part of the conversion to the improved nodal configuration that is described below.

5.2 Improved Nodal Computer Configuration

As reported in the previous SATS (see Reference [7]) a new nodal configuration was being considered as part of the plans to incorporate radio units into the DSN test bed. The new node configuration has now been specified in more detail and some of the necessary hardware has been procured and checked out.

The new configuration is shown in Figure 5-1. It is a multicomputer configuration with interprocessor communication through shared memory. The main processor boards are SUN processor boards and the system bus is the Multibus. There will be six such systems, one for each test bed node.

Other units on the Multibus include a parallel port card with handshaking, a serial port I O card and a Radio Unit Interface card. Two parallel ports will be used. One will be used to provide the interface between the Signal Processing Subsystem and the SUN board used for azimuth tracking. The other parallel port will be used to allow the DCU to reset any of the other processors in the node. Two serial ports are also required. One is to provide a



LEGEND:

RS	RESET	ATP	AZIMUTH TRACKING PROCESSOR
CP	CONSOLE PORT	PTP	POSITION TRACKING PROCESSOR
HP	HOST PORT	DCU	DIGITAL COMMUNICATION UNIT
		ECC	EXPERIMENT CONTROL COMPUTER

Fig. 5-1. Improved Nodal Configuration.

128181 N

control connection to Signal Processing System. The other will be the initial interface for any locally attached user processors. The Radio Unit Interface that provides the interface to the CNT Radio Units is described in Section 5.3.

The necessary SUN processor boards have been procured. Each board has a private bus that supports 256KB of RAM, two serial I/O ports, a five channel timer and a parallel input port but without handshaking. The 512KB shared memory boards and card cages with rack mounts have also been procured. One serial I/O board and one parallel I/O board have been procured for evaluation.

The first prototype of the new node configuration is under construction. One of the card cages has been equipped with a power supply and the necessary I/O connectors. The SUN boards and one 512KB memory board have been installed. This system has been attached to our PDP 11/70 developmental computer for initial evaluation. It is now being used to convert software and for further experimentation. Figure 5-2 is a picture of this partially constructed system.

5.3 Radio Unit Interface

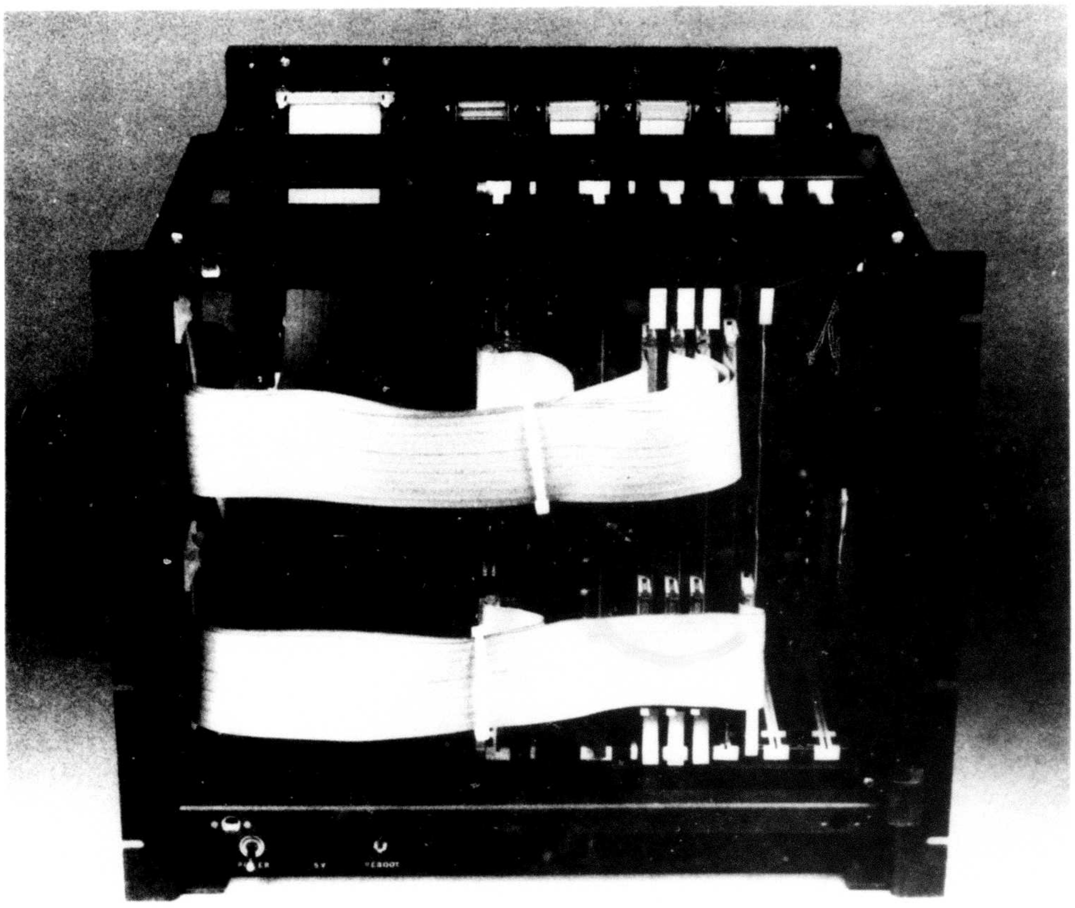
A Radio Unit Interface (RUI) will provide the interface between the multiple micro-computer DSN node and the Radio Unit (RU) being provided by the CNT project. It will include the data paths and the necessary control functions to and from the radio unit.

Figure 5-3 shows a block diagram of the RUI design. It contains the four major sections that are described below. They are the Multibus Slave Interface section, the FIFO I/O section, the Real Time Clock section and the 8089 Subsystem section.

The RUI is a slave device on the Multibus and must respond to commands sent by bus masters. The Multibus Slave Interface Section of the RUI provides this interface for bus masters. It will normally be used by the DCU for RUI initialization, loading of channel programs, and reading and writing of the real time clock.

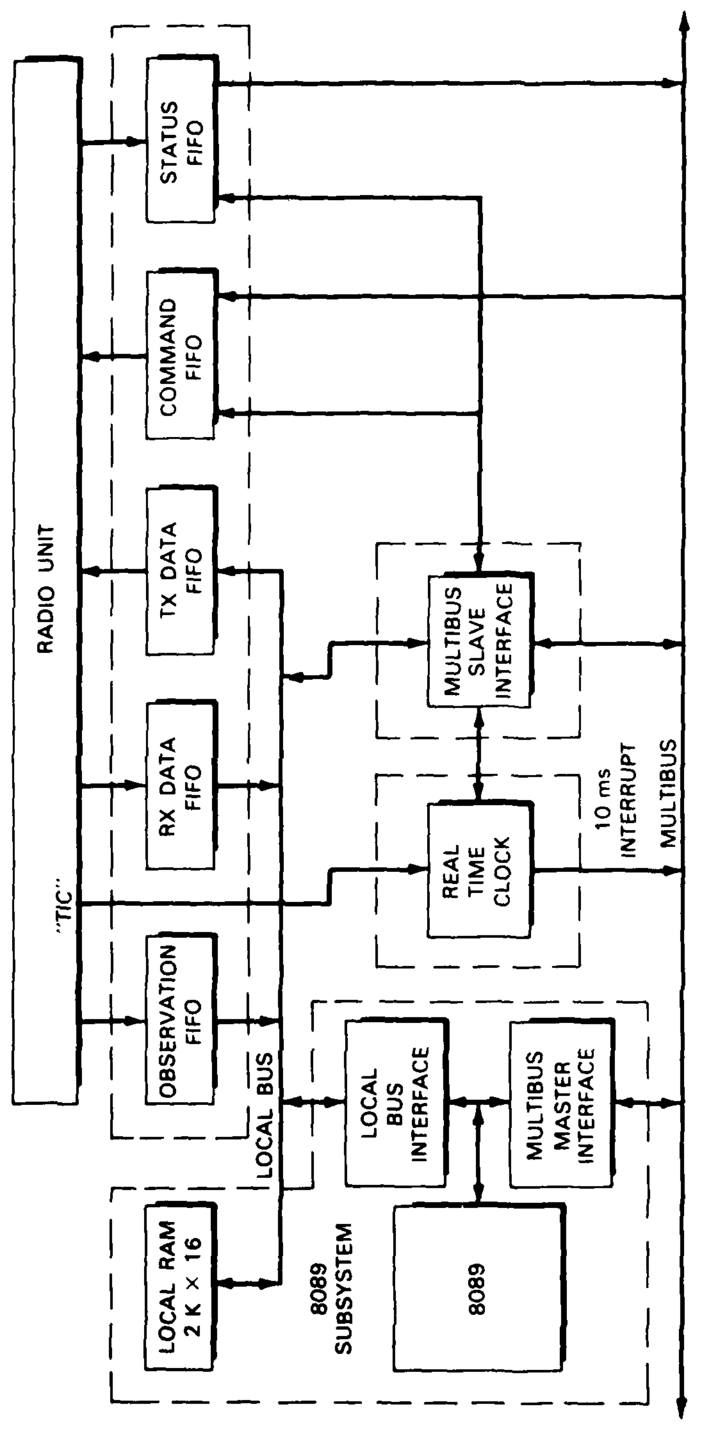
The FIFO I/O Section provides the interconnection between the RUI and the radio. The FIFOs provide the buffering, parallel to serial conversion and serial to parallel conversion of data and control information passed between the two units. They can be accessed by either the Local Bus or the Multibus.

The Real Time Clock section provides the timing information needed for node synchronization and self-location functions. It also provides the interrupt to the DCU from the 10 ms "tic" provided from the radio.



128182-N

Fig. 5-2. Prototype Node Under Construction.



128183-N

Fig. 5-3. Radio Unit Interface.

6. RADIO COMMUNICATION AND SELF-LOCATION SOFTWARE

6.1 Communication and Ranging Protocols and Software

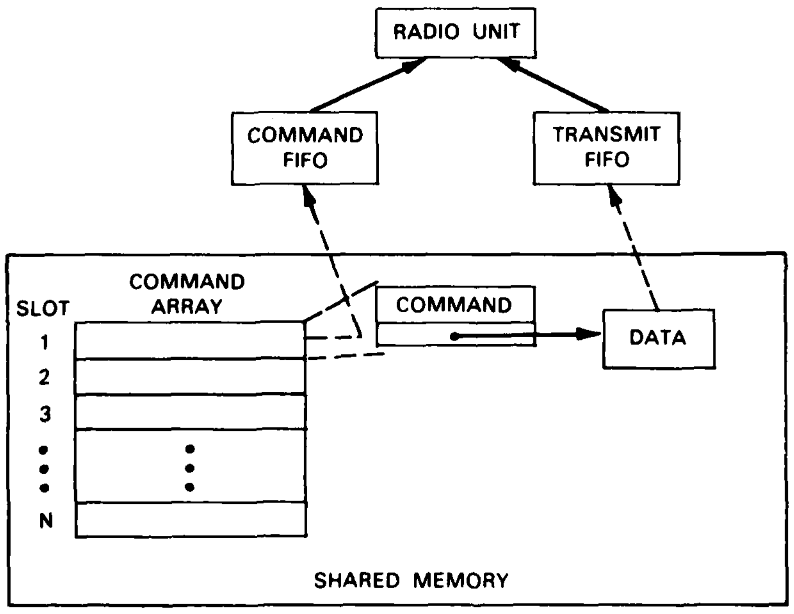
The basic digital portion of DSN test bed radios consists of a SUN processor, shared memory and a radio unit interface as discussed in Chapter 5. The basic software in the DCU and the 8089 will perform the transfer of commands, data and status to and from the radio. Three levels of software have now been specified for the radio system. These are:

1. The user program will provide the user interface for data transmission, data reception, status information and ranging information.
2. The DCU driver will translate user transmission requests into the appropriate command for the radio; respond to user data requests for messages received by the radio; translate internodal ranging requests into ranging protocol; and provide status to the user programs.
3. Data transfer software will control the transfer of data between the radio unit and shared memory. This software is split between the DCU's radio interrupt service routine and the 8089 channel program.

At the present time a primitive version of the DCU driver has been written and tested in a Unix environment with simulated data transfer to and from the radio.

The communication software must perform three major functions: data transmission, data reception and ranging. The following discussion outlines how the system is being designed to perform those functions using the three software layers. In all cases the user level treats the radio like any I/O device by opening a data stream in read, write, or read/write mode.

Figure 6-1 schematically shows how data transmission is accomplished. To transmit data to the radio, the user issues a call to a write subroutine. A pointer to the transmit data is passed to the subroutine which then copies the data into shared memory. The DCU driver responds to the subroutine call by formatting the appropriate command packet for the radio. A message header is added to the data with the data field set to indicate that there is user data within the packet. The driver examines its list of command packets scheduled for the next N slots and inserts the command into an available slot. When a tick arrives from the radio for that slot, the command is copied from the appropriate element of the command list in shared memory into the command FIFO. One of the two 8089 channels will control



128184-N

Fig. 6-1. Data Transmission.

the transfer of the transmit data from the shared memory to the transmit FIFO. This channel is activated by the 68000. When the next tick arrives from the radio, the 68000 driver will read the radio status from the status FIFO into shared memory to determine if the transmission was successful, unsuccessful or still in progress. The user can examine the status of the transmission.

Figure 6-2 shows how a reception takes place. On reception the radio fills an observation FIFO and a received data FIFO. The 8089 channel is always active and ready to receive data while there are free buffers. The 8089 processor's channel program copies data from the observation and received data FIFOs into the shared memory. It formats the observation and received data into a single structure. Then it links this structure onto a queue of received data which also resides in the shared memory.

Both data and ranging information appear on the received data queue. On the tick interrupt from the radio the 68000 driver examines the received data queue. It separates ranging and user data structures into two separate queues based on a header which indicates whether it contains ranging information or user data.

The user reads data by issuing a read subroutine call with a pointer to a buffer and a byte count as parameters. The driver in the 68000 responds to the read request with data from the data queue. The user can examine the observation associated with the data by issuing a subroutine call.

Ranging information will be treated like an I/O stream which the user can open in read, write, or read/write mode. Users can request ranging information from a specific node. The ranging request is issued by a subroutine call with a pointer to data containing source and destination node IDs. When the driver receives a range request, it invokes a ranging protocol. It copies the range source and destination IDs into a structure and indicates that this packet contains ranging information in the packet header. A command structure is set up and the driver queues the range request in an empty slot. Based on that slot number, it calculates the time of transmission and inserts it into the range data packet.

When the ranging request is received by the destination, the 68000 driver retransmits the range packet which was sent along with it at the time of arrival. The driver will set up a command structure for the ranging reply and queue it into an empty slot. Again, the time of transmission is calculated based on the slot number and is inserted into the packet.

The range reply packet is received by the requesting node and the driver responds with an acknowledgement containing the packet it received along with time of arrival and time of

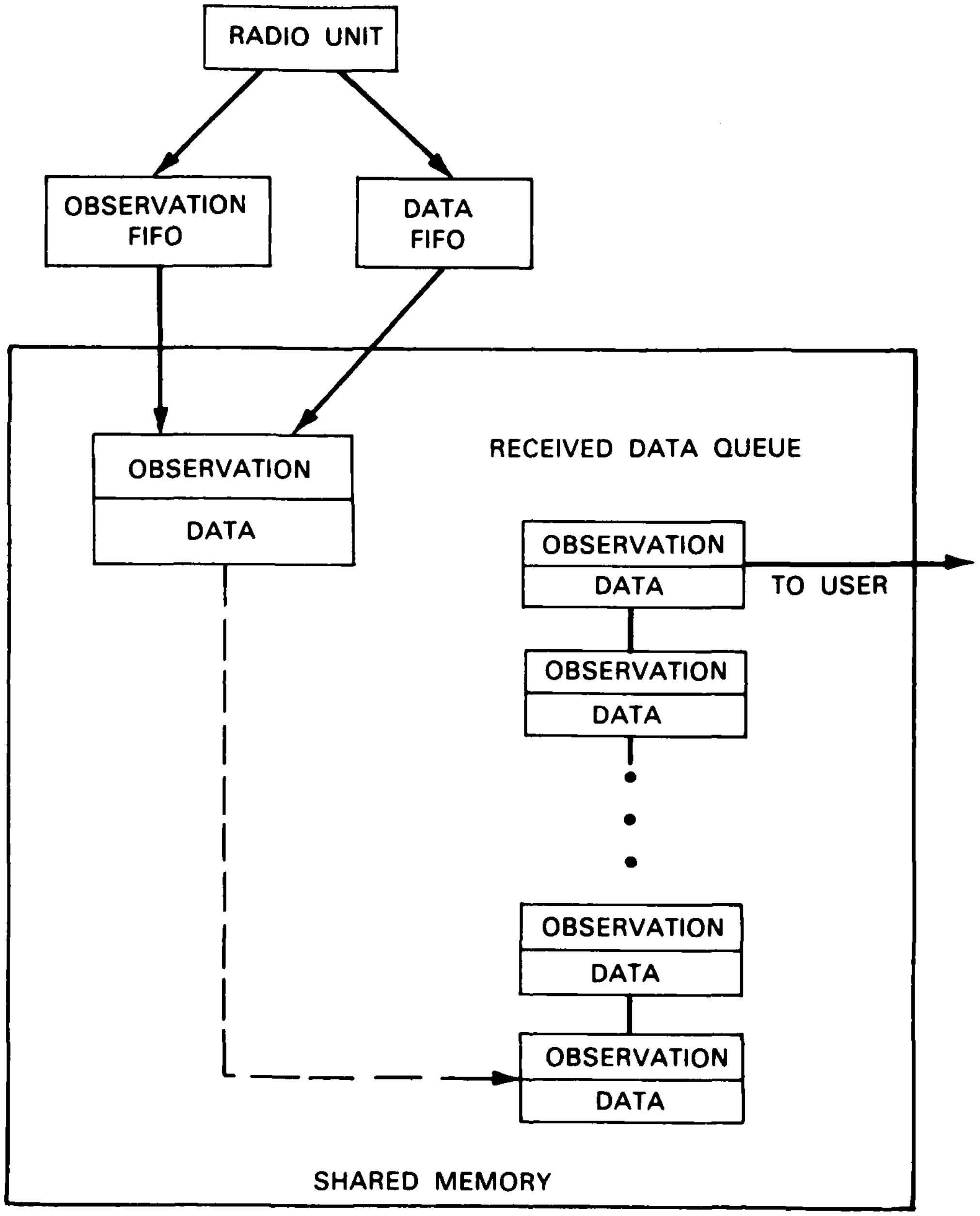


Fig. 6-2. Data Reception.

128185 N

transmission. If the clocks slew during this operation, the amount of the slew must be included. This acknowledgement is passed to the destination node so that it can also obtain the range measurement.

The user will examine the range data returned by the destination by issuing a subroutine call with a pointer to a buffer and the number of bytes to transfer as parameters. The driver processes the subroutine call with data from the range queue. The observation packet associated with the ranging measurement can also be examined.

6.2 Self-Location Software

During this reporting period we worked with a team from Columbia University to install a version of their position location software on the test bed. This software contained an artificial data generator, which generated ranges between the nodal radio units and the algorithms, which determined the locations of the nodes from these ranges. The software was first installed on our 11/70 computer and demonstrated to run under a conventional Unix environment. We then converted the software to run using fixed point arithmetic and message based I/O. Finally we downloaded the software into a testbed node and demonstrated that it worked on some simple test cases. This activity validated that there are no fundamental software incompatibilities between the Columbia effort and the test bed.

REFERENCES

1. Distributed Sensor Networks Semiannual Technical Summary Report, M.I.T. Lincoln Laboratory, 31 March 1978, p. 40.
2. Distributed Sensor Networks Semiannual Technical Summary Report, M.I.T. Lincoln Laboratory, 31 March 1980, p. 6.
3. Van Trees, H., "Detection, Estimation and Modulation Theory," Part I, John Wiley, 1968, p. 247-271.
4. Therrien, C., "Discrimination of Random Delay Modulated Signals with Application to Target and System Identification," Report, Naval Postgraduate School, EE Dept., Monterey, CA., September 1982, p. 11-23.
5. Widrow, *et al.*, "Adaptive Noise Canceling, Principles and Applications," Proceeding IEEE, Vol. 63, No. 12, December 1975, p. 1692-1716.
6. Haubrich, R.A., "Array Design," Bulletin of the Seismological Soc. of Amer., Vol. 58, No. 3, June 1968, p. 977-991.
7. Distributed Sensor Networks Semiannual Technical Summary Report, M.I.T. Lincoln Laboratory, 31 March 1982, p. 21.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-83-020	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Distributed Sensor Network		5. TYPE OF REPORT & PERIOD COVERED Semiannual Technical Summary 1 April 1982 — 30 September 1982
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard T. Lacoss		8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element Nos. 61101E, 62708E ARPA Order No. 3345 Project Nos. 3D30 and 3T10
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE 30 September 1982
		13. NUMBER OF PAGES 58
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Electronic Systems Division Hanscom AFB, MA 01731		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

multiple-sensor surveillance system	acoustic sensors
multisite detection	low-flying aircraft
target surveillance and tracking	acoustic array processing
communication network	digital radio

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the work performed on the DARPA Distributed Sensor Networks Program at Lincoln Laboratory during the period 1 April 1982 through 30 September 1982.

DD FORM 1 Jan 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)